The Dissertation Committee for Elias Franchot Ponvert
certifies that this is the approved version of the following dissertation:

Unsupervised Partial Parsing

Committee:

Jason Baldridge, Supervisor

Colin Bannard

David Beaver

Katrin Erk

Raymond Mooney

Unsupervised Partial Parsing

by

Elias Franchot Ponvert, BA, MA

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

August 2011

To Adèle, Antonio and Drucilla, who did not get to see this,
and to Jasper and Katharine, who got here just in time.

Acknowledgments

I am indebted to many individuals for the help they have given me in the creation of this work. First and foremost, my adviser, Jason Baldridge, provided the motivation, inspiration, and the tools I needed to develop as a student and researcher. Jason turned me on to the problem of parser induction, and shared his enthusiasm for the application of machine learning to the challenges of natural language. Katrin Erk, my unofficial co-adviser, also contributed enormously to my growth as a computational linguist researcher, and to this work in particular. Katrin and Jason bring distinct perspectives and interests to their study of language, and this diversity of perspective very certainly benefited this research. I thank each of the other members of my dissertation committee – Colin Bannard, David Beaver and Ray Mooney. Colin consistently asked the best and most challenging questions, while all along was as supportive and encouraging as anyone. David met this work with an excitement he brings to all sort of natural language research; he raised dozens of insightful questions, provided dozens more sharp and thoughtful suggestions. And, I am indebted to Ray, not just for the encouragement, ideas, and suggestions – though I am for these, too. As much as anyone in computational linguistics, Ray has championed strict empirical evaluation of ideas, and he has advocated dropping spurious and unrealistic assumptions; without a doubt, these ideals directly inspired the development of the research in this thesis.

It has been a wonderful experience to work in the multidisciplinary commu-

nity of computational linguistics researchers at the University of Texas at Austin. For comments, questions and advice contributing to this research, I thank Yinon Bentor, Hans Boas, Christopher Brown, Travis Brown, David Chen, Weiwei Ding, Joey Frazee, Dan Garrette, Leif Johnson, Igor Karpov, Taesun Moon, Joe Reisinger, Mike Speriosu, Ben Wing and Andrew Young; and especially Matt Lease, who opened my eyes to a wide range of ideas during the early stages of this research. Also, this research has benefited from conversations I have had with scholars visiting our group at UT, and others I connected with at the 2010 International Conference on Semantic Computing and the 2011 meeting of the Association for Computational Linguistics; I owe particular thanks to David A. Evans, Yoav Goldberg, Mark Johnson, Dan Klein, Kevin Livingston, Fergal Monaghan, Graham Neubig, Ted Pedersen, Philip Resnik, Noah Smith, and Valentin Spitkovsky. Finally, Yoav Seginer's work is an inspiration to this research, and I thank Yoav for comments and feedback on his work and on this. Suffice to say, any errors remaining in this work are my own.

The last six years of graduate study have been extremely stimulating and enjoyable, and I thank my teachers Jason Baldridge, David Beaver, John Beavers, Megan Crowhurst, Katrin Erk, Scott Myers, Harvey Sussman, and Stephen Wechsler for all I have learned from them. With sadness, I also mention Carlota Smith, an inspiring teacher, supervisor and mentor, who passed away in 2007. Finally, Richard Meier has done a fantastic job as department head, and I thank for all he has done for our research community.

The students of the department of linguistics are a great group, and I am pleased and honored I have gotten to spend the time I did with them. Special thanks go to the members of my dissertation writing group: Emilie Destruel, Nick Gaylord, and Tony Wright. I thank Nick, especially, my officemate for the last two years, for talking me out of the various conceptual, rhetorical and emotional ruts I

got into during the writing of this thesis. And, I owe deep and thorough thanks to the staff of the department of linguistics, Leslie Crooks, Gina Pollard and Benjamin Rapstine for all they have done for me at every stage in my graduate career.

I thank my beautiful wife, Candace Pruett, for her patience, encouragement, humor, and most of all her love during the stress and long hours of the graduate school experience. She and our children, Katharine and Jasper, made the experience worth it – if, for nothing else, the joy to come home every night to a quirky, cute, fun and loving family. Lastly, for all she did to raise me right and set me on a good path, I thank my mother, Annie Duncan-Ponvert. Thanks, Mom.

ELIAS FRANCHOT PONVERT

The University of Texas at Austin
August 2011

Unsupervised Partial Parsing

Publication No. _____

Elias Franchot Ponvert, Ph.D.
The University of Texas at Austin, 2011

Supervisor: Jason Baldridge

The subject matter of this thesis is the problem of learning to discover grammatical structure from raw text alone, without access to explicit instruction or annotation – in particular, by a computer or computational process – in other words, *unsupervised parser induction*, or simply, *unsupervised parsing*.

This work presents a method for raw text unsupervised parsing that is simple, but nevertheless achieves state-of-the-art results on treebank-based direct evaluation. The approach to unsupervised parsing presented in this dissertation adopts a different way to constrain learned models than has been deployed in previous work. Specifically, I focus on a sub-task of full unsupervised partial parsing called *unsupervised partial parsing*. In essence, the strategy is to learn to segment a string of tokens into a set of non-overlapping constituents or *chunks* which may be one or

more tokens in length. This strategy has a number of advantages: it is fast and scalable, based on well-understood and extensible natural language processing techniques, and it produces predictions about human language structure which are useful for human language technologies. The models developed for unsupervised partial parsing recover base noun phrases and local constituent structure with high accuracy compared to strong baselines.

Finally, these models may be applied in a cascaded fashion for the prediction of full constituent trees: first segmenting a string of tokens into local phrases, then re-segmenting to predict higher-level constituent structure. This simple strategy leads to an unsupervised parsing model which produces state-of-the-art results for constituent parsing of English, German and Chinese. This thesis presents, evaluates and explores these models and strategies.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The subject matter of this thesis is the problem of learning to discover grammatical structure from raw text alone, without access to explicit instruction or annotation – in particular, by a computer or computational process – what I and others (Seginer, 2007a,b; Reichart and Rappoport, 2010) call *unsupervised parser induction*, or simply, *unsupervised parsing*.

In this work, I will present a method for raw text unsupervised parsing that is simple, but nevertheless achieves state-of-the-art results on treebank-based direct evaluation. The approach to unsupervised parsing presented in this dissertation adopts a different way to constrain learned models than current research in the field. Specifically, I focus on a sub-task of full unsupervised parsing called *unsupervised partial parsing*. In essence, the strategy is to learn to segment a string of tokens into a set of non-overlapping constituents or *chunks* which may be one or more tokens in length. In this dissertation, I will show that this strategy has a number of advantages: it is fast and scalable, based on well-understood and extensible natural language processing techniques, and it produces predictions about human language structure which are useful for human language technologies. The models developed for unsupervised partial parsing recover base noun phrases and local constituent structure with high accuracy compared to strong baselines.

Finally, these models may be applied in a cascaded fashion for the prediction of full constituent trees: first segmenting a string of tokens into local phrases, then re-segmenting to predict higher-level constituent structure. This simple strategy leads to an unsupervised parsing model which produces state-of-the-art results

for constituent parsing of English, German and Chinese. This thesis presents, evaluates and explores these models and strategies.

1.1 Motivations

Before expanding further on this proposal, I will outline some of the motivation for studying human language parsing in general (§1.1.1), and for studying it from an unsupervised perspective in particular (§1.1.2).

1.1.1 Why study human language parsing?

It is a foundational tenet of theoretical linguistics that human understanding of natural language is mitigated in part by structural, or syntactic, processing of linguistic information.

Human language parsing can mean a number of things in different contexts; here what is meant is the development of computer programs (or, more generally, processing), that can provide some representation of syntactic structure for sentences. The form these representations take, for the purposes of this dissertation, is discussed in Chapter 2. Suffice to say, it is hardly likely that any single (or finite) representation of syntactic structure can be considered a *correct* representation of how a speaker understands syntactic structure. Nevertheless the representations provided by linguists for sentences in *treebanks* – collections of annotated sentences – will be taken as a sufficiently correct representation for evaluation.

So, why study human language parsing? The main motivation of the present work is the hope that efficient and accurate replication of human processing of text into syntactic structure may contribute to the development of useful *human language technologies.*

Simple techniques of human language processing have dominated the most successful human language technologies for years, for example N-gram language modeling or bag-of-words representations of documents for document classification. However, the changing landscape of human language technologies requires more sensitive representations of natural language than had previously been employed. For instance, Yamada and Knight (2001) use syntax and parsing in a machine translation model; Punyakanok et al. (2008) use syntax and parsing infor-

mation in a semantic-role labeling system; and Bergsma and Lin (2006) use parsed syntactic structure as a core component of a system for pronoun resolution. In addition, potential applications of the predicted grammatical structure to concept acquisition (Davidov et al., 2009), search query segmentation (Bendersky et al., 2009), and text classification such as *geolocation* (Eisenstein et al., 2010; Wing and Baldridge, 2011) are discussed in what follows, just to name a few.

1.1.2 Why study unsupervised parsing?

Much parsing research is from a *supervised* perspective. That is, a treebank is used as *training material*, where the linguists' annotations as to syntactic structure are used as input to a machine learning process, the end result of which is a computer program which can replicate syntactic structure annotations on new (unseen) sentences.

In general, supervised parsers outperform unsupervised parsers dramatically in direct evaluation. Moreover, they can make certain kinds of predictions – constituent node labels, labeled grammatical dependencies, *etc.* – that unsupervised parsers cannot simply because this information is available directly or indirectly in the training material, whereas it is not in an unsupervised setup.

However, these limitations only apply when there is training material available. When there is no training material available, then a supervised approach cannot be used. This is obviously the case when approaching a totally new language. This is also the case when the textual domain is radically different from what may be available in annotated training, even if nominally in the same language. This is the case with social media text, such as e-mails and user content on social networking applications (Twitter, Facebook, Gowalla, *etc.*). Supervised parsers are actually relatively good at *domain adaptation* – that is, applying models trained on data in one textual domain to new data from another (Hollingshead et al., 2005). But social media text provides challenges that are beyond the kind of differences usually encountered moving between textual domains.

An example of an application of unsupervised parsing is given by Davidov et al. (2009). They use an unsupervised parser to provide syntactic/structural features to improve concept acquisition. The system they use (Seginer's (2007) CCL parser, which is described below) specifically matches well with this goal since a)

it is fast and memory efficient, thus can be applied to the large datasets they work with, and b) it is unsupervised and operates on raw text, thus it can be applied with minimal data preparation to texts in Russian, for which there are relatively few trained linguistic models.

To sum up, the primary motivation for parsing research, in this work, is to contribute to human language technologies. The primary motivation for unsupervised parsing research, then, is to learn parsers for new languages and out-of-domain text, to contribute to human language technologies. With these in mind, an outline of the ideal properties an unsupervised parsing system will have is given here.

1.2 Desiderata for an unsupervised parser

Given the motivations outlined above, here are the properties of an ideal solution to the unsupervised problem.

1. Operate on raw text. To as much of an extent as possible, an unsupervised parser should operate on actual textual data, with limited modifications and simplifications. As will be discussed in §2.3.1, many contemporary unsupervised parsing methods assume the input to be a sequence of hand-annotated part-of-speech (POS) tags, rather than actual terms. Obviously, such an assumption makes deployment of parsing technology to completely new languages and datasets impossible, if there are not annotated POS annotations available. For example: it is only now, with the work of Gimpel et al. (2011), that any POS resources are available for Twitter social media text; and only for English posts (tweets). This matter is discussed in §2.3.1, and explored in following chapters. Related to this, an unsupervised parser should perform adequately well on all sentences – unsupervised parsers have frequently only been evaluated using 10-word-or-less sentences. Especially for textual domains like news-text, this leaves the vast majority of sentences out of consideration.

2. Extensible. Several recent unsupervised parsing systems (*e.g.* Seginer, 2007a; Hänig, 2010) make use of a complicated sets of rules and heuristics. While these may work well for the evaluation provided, the techniques presented are difficult to understand and replicate. Thus, the performance in evaluation

is difficult to justify, and future research which could build on the presented techniques is basically cut-off. An ideal unsupervised parsing system would incorporate learning and processing techniques which are applications of existing natural language processing techniques, or straightforward extensions thereof.

3. **Efficient.** To facilitate deployment to large datasets – which is a requirement for most human language technologies, now – an unsupervised parsing system must be able to produce results efficiently, with respect to computation time and space (memory) required.

4. **State of the art performance.** Finally, satisfaction of the above considerations should not come at the expense of performance. Even while striving to achieve the goals above, new research in unsupervised parsing should (of course) improve on existing techniques in empirical evaluation.

1.3 Contributions

A primary contribution of this research is that unsupervised parsing satisfying the desiderata outlined above *is possible*. The strategy presented in this work which achieves this does so by generalizing on a different learning problem: *unsupervised partial parsing* (UPP). What this means is that, rather than learning a model to predict trees directly, the task is to learn to segment sequences of terms into multiword constituents. For example, from the string

the quick brown fox jumps over the lazy dog,

a UPP strategy which accurately identifies noun phrases would output two constituents for this sentence:

(the quick brown fox) jumps over (the lazy dog).

UPP is defined as: *the unsupervised segmentation of strings of terms into non-overlapping grammatical constituents.* This is meant as the unsupervised analog of supervised NP identification or chunking (Abney, 1991; Ramshaw and Marcus, 1995; Tjong Kim Sang and Buchholz, 2000), and is often here referred to as *unsupervised chunking*. While the predicted level of representation is only local in scope, standard unsupervised sequence models (hidden Markov models with expectation-maximization

training) can learn to predict local constituents with high accuracy (Chapter 3). Moreover, chunking models can be coordinated in a *cascade of chunkers* to produce full constituent tree structures with state-of-the-art accuracy (Chapter 4).

In passing, the unsupervised chunking models and evaluation are a significant contribution in their own right. As far as I know, this problem has never been proposed as such in NLP research, and never connected to unsupervised parsing, as here. Direct evaluation for this task is proposed as subsets of treebank constituent annotations. Two such evaluations are proposed, representing different views of local treebank constituents: *constituent chunks* (lowest level multiword constituents) and *base NPs* (non-nested noun phrases).

1.4 Plan of the dissertation

This dissertation is organized as follows. In Chapter 2 I present the central problems addressed in this work, unsupervised parsing and unsupervised partial parsing. An outline and brief overview of unsupervised parsing follows, and a discussion of the methodology adopted in this research, which centers around empirical evaluation via comparison with human annotated linguistic resources. The evaluation is described for both problems addressed, and the datasets used in evaluation.

Chapter 3 presents some of the core contributions of this research, probabilistic sequence models for unsupervised partial parsing. Evaluation with respect to treebank annotations comes next, together with detailed results and model analysis. This includes learning curves, experiments using additional data that what is provided in the standard treebank training set, and direct inspection of the model parameters being learned. Finally, I consider some variations on the model, and provide a brief discussion of the computational complexity involved.

Chapter 4 presents the other main contribution of this work: the use of the unsupervised partial parsing models described in Chapter 3 to produce full parse trees by using a cascade of partial parsers. This strategy is articulated and evaluated via comparison to parsing baselines, and two benchmark systems, a raw text unsupervised parser and a gold-standard POS based system. The proposed parsing strategy is discussed and analyzed, similarly to the discussions of the partial parsing strategies before. Chapter 5 concludes and suggests future directions for this research.

Chapter 2

Tasks, methodology and data

This chapter outlines the methodology applied in this research. I begin by reviewing the general problem addressed in this work: unsupervised parsing, and provide a general discussion of nature of this task. Then, the methods I use are discussed: this includes the focus on experimental methods, and the usage of annotated resources as a basis for evaluation. The specific tasks addressed in this work are reviewed: unsupervised constituent parsing and, a subtask of unsupervised parsing, unsupervised chunking. Finally, the specifics of a task are defined by its evaluation; the evaluations for the tasks described conclude this chapter.

2.1 Unsupervised parsing

The problem of unsupervised parsing is closely related to that of learning a *grammar* from text directly, or *unsupervised grammar induction* (UGI), which has a relatively long history within the relatively brief history of computational linguistics. However, in this work, I shy away from referring to this or related research as grammar induction, since the object of UGI is to learn a *grammar*, which I take to be a probability distribution over grammar rules. Indeed, early approaches (Lari and Young, 1990) to UGI did, in fact, aim to learn a probabilistic context-free grammar (PCFG), which is a probability distribution over context-free rules, familiar from generative grammar (Chomsky, 1957). I feel this perspective on the problem is too limiting, as it seems to presuppose the solution at some level. Rather, the problem is to learn is a *program* or *process* for predicting grammatical structure in novel text

– for this reason I use the term *unsupervised parsing* throughout this work.

At a basic level, the difference between *unsupervised* methods of natural language processing and *supervised* methods is the availability of supervision, which usually takes the form of *annotated training material*. What this means is that a corpus of natural language texts – a collection of newspaper articles, for instance – are provided with human-made annotations of linguistic (or other information). An early example of such annotations comes from the 1979 version of the Brown Corpus (Francis and Kucera, 1964) which provided part-of-speech (POS) annotations for each word in the corpus documents. A little over a decade later, the Penn Treebank (Marcus et al., 1993, 1999) was released, which provides a much richer set of linguistic annotations. Not only are part-of-speech annotations provided, but the grammatical *structure* of each sentence is given (as a tree-structure, *c.f.* §2.3.1). Resources such as these – the Brown Corpus, the Penn Treebank, and others – also provide implicitly or explicitly other linguistic information, notably sentence boundaries are where words begin and end (tokenization). Other corpora may contain annotations which may only loosely be considered linguistic or grammatical information. For instance, sentiment corpora (*e.g.* Constant et al., 2009) are annotated for the sentiment (*positive, negative, neutral*) or opinion of the author of the text.

However, what is meant by supervised methods in NLP is not just methods which operate on annotated text. A large body of research in unsupervised parsing, for instance, makes a crucial assumption that human-annotated part-of-speech annotations are available. Rather, supervised methods using annotated resources are intended to *learn a program which can replicate the annotation process on new material*. For instance, a POS tagger is a program which assigns POS annotations to words in context; a supervised POS tagger, which learns from annotated material, is typically designed to produce predictions on new material which best represent the assumptions and insights which the annotators applied when they assigned POS tags to the words of the texts in the training material. Another way of thinking of this is, the output of a supervised POS tagger is intended to best approximate the annotations the human annotators would have assigned to the new material.

By contrast, *unsupervised methods* in NLP are also intended to discover or produce annotations which correspond to linguistic structure or patterns, however they do not have access to exemplars of said linguistic patterns. Again, thinking

of POS tagging, an unsupervised POS tagger may have access to text to learn patterns of sequences of words, but not to the POS annotations; nevertheless, typically such a system is *evaluated* by the degree to which its predicted output corresponds to the human-made annotations on some evaluation dataset. Having said that, almost never is such a system provided *no* linguistic information, other than the sequence of characters of the text – for instance, usually such a system will have access to word-boundaries (tokenization), and sometimes sentence boundaries and document boundaries. Good tokenization is not always easy: for some languages, such as Chinese, Japanese and Korean, the written language does not make use of white-space or punctuation to indicate word-boundaries, as do most Western scripts. Indeed, there are supervised and unsupervised approaches to tokenization and sentence identification as well.

I raise these points, firstly, to distinguish supervised and unsupervised methods in natural language processing, as this dissertation will focus on the latter. But, also, this discussion underlines the point that even unsupervised methods make some assumptions about what information, beyond the character sequences of the original digital texts, is available in the proposed learning methodology. And this raises a question: how much information, assumed given in the natural language processing task, is too much? And that question must rely on a larger motivational question: why study the proposed methods at all?

For instance, if the scientific goals of unsupervised parsing is to model human language acquisition, then the information available to the computer program should be intended to represent the kind of information available to a human language learner. On the other hand, if the general research goal is to advance human language technologies, then the information assumed should reflect the kind of information available in a practical technological setting – e.g. during the process of harvesting electronic documents for textual analytics and data mining.

In this dissertation, I suggest that among the assumptions standardly made in unsupervised partial parsing research, a few stand out as unrealistic given most of the motivations of the research. Chief among these is the assumption that perfect knowledge of human-made annotations for parts-of-speech (POS) are available for parser learning and evaluation. Here and later this is refereed to as the gold-standard part-of-speech (GPOS) assumption. In contrast to GPOS-based parser learning, I propose that *raw text unsupervised parsing* is a more appropriate goal

for unsupervised parsing research, at least for the purpose of advancing practical human language technologies research.

Of course, it is not for nothing that researchers make simplifying assumptions – such as access to GPOS. Unsupervised parsing systems, in particular, aim to learn and predict very specific structures based only on observed strings of symbols. To think of parsing as a searching problem, the search space of possible analyses (parse-trees, in this work) over a given string of symbols is very large. Predicting the correct analysis over a string is very difficult, and unsupervised model-learning methods, such as the expectation-maximization methods of Klein and Manning (2002, 2004), can easily wander away from learning models whose predictions correlate with human language structures. Simplifying assumptions, like learning from GPOS rather than raw text, in addition to structural bounds and constraints, typically manifested in learning only from short ($\leq$10-word sentences), adequately constrain learned models to make reasonably accurate predictions.

2.2 Methodology

The goals of this research are to find computational methods to discover the latent grammatical structure in natural language texts with minimal data-preparation or human annotation. The methodology employed is the standard for empirical computational sciences: methods (algorithms) which may achieve the desired goals are proposed, the proposed methods are implemented as a concrete computer program, the program is evaluated by analyzing its output using known collections of data – here, texts.

Since the task (the problem) is the identification of latent grammatical structure, the textual material used for evaluation are texts which have been *annotated* for grammatical structure. This means that, ultimately, the aim of the proposed methods is to replicate human judgments – indeed, trained linguists' judgments – about linguistic grammatical structure.

2.3 Tasks

Two basic tasks are the subject of this work. The first – unsupervised (constituent) parsing – has a long history within the field of natural language processing. The

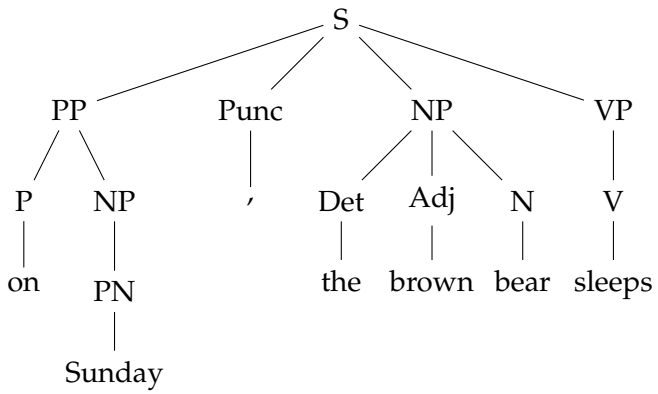


Figure 2.1: Basic constituent tree with labeled nodes.

second – unsupervised partial parsing – is a distinguished sub-task of unsupervised constituent parsing, which in this dissertation is promoted to a distinct task on its own, though with connections to the overall goal of general unsupervised parsing. In this section, both tasks are articulated in terms of their goals and experimental evaluation. Both tasks make underlying assumptions about the nature of the grammatical structure being predicted, and these assumptions impact evaluation, as well as the strategies available to solve them.

2.3.1 Unsupervised constituency parsing

An example of a basic constituent tree is illustrated in Figure 2.1. This kind of structure is familiar to most students of linguistics as a *phrase-structure tree* or, to use classic terminology, a *phrase-marker*. In classic generative grammar (Chomsky, 1957), this sort of structure was used to partially illustrate the derivational process which generated a string (*i.e.* a sentence) from a context-free grammar. For instance, using the example from Figure 2.1, the grammar would have at least the context-free rules illustrated in Figure 2.2. These rules characterize a set of strings of symbols (*i.e.* sentences). Moreover, they characterize a set of derivations, represented partially by corresponding tree structures: each non-terminal node of the phrase-structure tree corresponds to a rule application. For instance, the root node S corresponds with an initial application of the rule expanding the S non-terminal symbol. As a trace of the context-free grammar derivation, the phrase structure

				Prep	→ on
S	→	PP Punc NP VP		PN	→ Sunday
PP	→	Prep NP		Punc	→ ,
NP	→	PN		Det	→ the
NP	→	Det Adj N		Adj	→ brown
VP	→	V		N	→ bear
				V	→ sleeps

Figure 2.2: Rules corresponding to the phrase structure tree in Figure 2.1. The rules in the right column indicate *parts-of-speech* (POS).

($_{S}$ ($_{PP}$ ($_{Prep}$ on) ($_{PN}$ Sunday)) ($_{Punc}$,) ($_{NP}$ ($_{Det}$ the) ($_{Adj}$ brown) ($_{N}$ bear)) ($_{VP}$ ($_{V}$ sleeps)))

Figure 2.3: Bracketing representation of the constituent tree from Figure 2.1.

tree of Figure 2.1 is only a partial trace, since it does not indicate the order of rule-applications.

Another perspective on constituent structure is that the subtrees in Figure 2.1 – the constituents – correspond to legitimate grammatical sub-units of the sentence. Such sub-units often correlate with (compositional) semantic components of the sentence, and there is an assumed relationship between the phrasal type and the type of semantic component. For instance, noun phrases (NPs) are usually entity-denoting, or generalized quantifiers; verb phrases (VPs) often denote predicates or events; prepositional phrases (PPs) often indicate properties of events or entities, *etc.*. Another way of visualizing constituent structure is *bracket notation*, as in Figure 2.3, which is often how constituent structure is actually annotated in constituent-based treebanks such as the Penn Treebank.

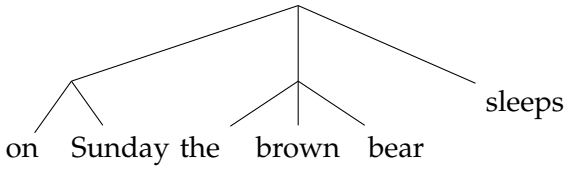
The notion of constituency – whether or not a given string of words does constitute a grammatical sub-tree – has a history in linguistics. Valid constituents, according to textbooks such as (Radford, 1988, p. 90), can be identified via classic *constituency tests*. Quoting Radford's textbook, here are some:

a. Does it [a sequence of words] have the same distribution as (i.e. can it be replaced by) an appropriate phrase of a given type? ...

b. Can it undergo *movement* (i.e preposing or postponing) ...

c. Can it serve as a *sentence-fragment*? ...

The task of unsupervised constituent parsing is to predict bracket structure



((on Sunday) (the brown bear) sleeps)

Figure 2.4: Unlabeled tree and bracketing for the example from Figure 2.1.

such as in Figure 2.3 given only a corpus of strings of symbols. Several factors characterize how this problem is specifically defined: how large a corpus, which symbols are taken to represent the units of language, *etc.*

One aspect of the unsupervised parsing problem is exactly how much of the full constituent structure is predicted by parsing models. In particular, the *joint learning* of bracket structure and bracket labels (PP, NP, VP *etc.* in the example above) is basically out of the scope of most contemporary unsupervised parsing models. Supervised models have access to bracket labels as part of the training material, and are usually evaluated on their prediction of brackets and their labels. This is natural for many supervised parsing models, since many learn to parse by learning probabilistic context free grammars (PCFGs) – basically a probability distribution over rules such as those in Figure 2.2. However, unsupervised approaches do not have access to these labels, so their direct prediction would be impossible from text alone.

Classic PCFG induction is hard – maybe impossible, if the target is a full natural language grammar – without informed constraints or other information (Lari and Young, 1990; Carroll and Charniak, 1993). For this reason, unsupervised parsing research often avoids the joint learning of constituent structure and constituent labels. In other words, the target of many unsupervised parsing methods is simply to predict the grammatical *structure* of sentences, so the final output would be a bracketing (or tree) as in Figure 2.4. The converse is also possible: Reichart and Rappoport (2008) learn to label unlabeled brackets. In particular, they learn to cluster constituents based on their content and context, and evaluate by checking the frequency with which the clusters overlap with treebank labels. They use predicted structures, produced by a separate system (the common cover links, or CCL, parser of Seginer, 2007a, which is discussed below).

- (the cat) in (the hat) knows (a lot) about that
- (the cat) (in the hat) knows (a lot) (about that)
- (the cat in the hat) knows (a lot about that)
- (the cat in the hat) (knows a lot about that)
- (the cat in the hat) (knows a lot) (about that)

Figure 2.5: Some possible segmentations for a simple sentence.

The example in Figure 2.4 also illustrates another convention standardly adopted by unsupervised parsing systems with respect to output and evaluation: punctuation is usually dropped from system output, and ignored during evaluation.

2.3.2 Unsupervised partial parsing

As alluded to in the introduction, one of the central proposals of this research is an alternative view of unsupervised constituent parsing, via a sub-problem focused on local constituent prediction: unsupervised partial parsing. This means the segmentation of a sequence (a sentence) into non-overlapping sub-sequences.

Constituent trees are hierarchical: longer brackets strictly contain smaller brackets. The set of bracketings b over a sentence $\mathbf{x}$ is just those bracketings with no crossing brackets. Say a bracketing b consists of a set of pairs $\langle i, j \rangle$ of indices of words in $\mathbf{x}$; so for each bracket $\langle i, j \rangle \in b$ where b is a bracketing for $\mathbf{x}$, $1 \leq i < j \leq |\mathbf{x}|$ ($|\mathbf{x}|$ is the length of the sequence $\mathbf{x}$). Moreover, a bracket cannot cross another in b, so $\langle i, j \rangle \in b$ only if for no $i' < i < j' < j$ is $\langle i', j' \rangle \in b$; likewise for no $i < i' < j < j'$. Unsupervised chunking targets a subset of bracketings over a string. In unsupervised partial parsing, the goal is to *segment* a string $\mathbf{x}$ into non-overlapping subsequences s. So, if $\langle i, j \rangle \in s$ and $\langle i', j' \rangle \in s$, then either $j' < i$ or $j < i'$.

Figure 2.5 illustrates with several different unlabeled segmentations of a simple sentence. However, this example also alludes to a problematic aspect of the task as defined: *which* segmentation is to be targeted? Certainly, any non-overlapping subset of grammatical constituents is too broad: each of the constituents

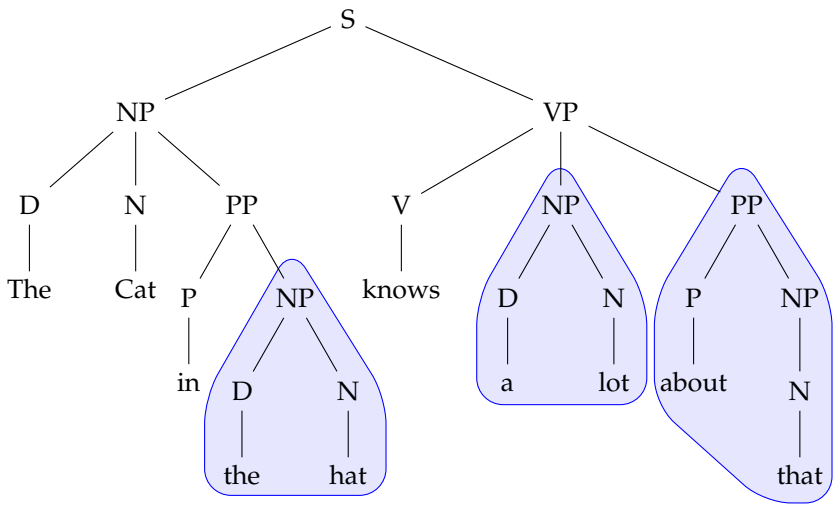
in the examples in Figure 2.5 is defensible as grammatical.

Of course, unsupervised parsing has a similar issue: which bracketings are ultimately the target of the parser learning process? The intuition is that full grammatical constituent trees are the target of unsupervised parsers, but ultimately the evaluation defines the task: the trees from gold-standard treebank annotations which are used to evaluate parser output are the de facto target of learning.

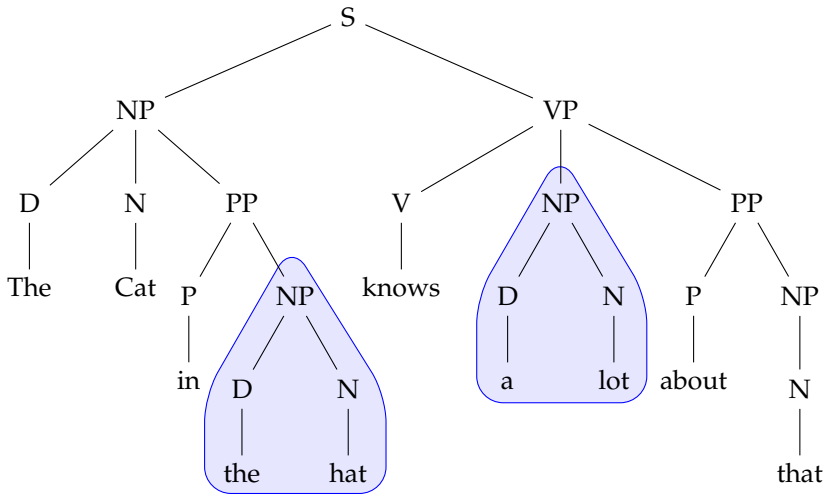
For unsupervised partial parsing, there are basic motivations, and direct evaluation based on treebank annotations represents those motivations. One motivation is to provide local constituent predictions as a starting point for unsupervised parsing: parsing proceeds by building larger constituents on top of the original local ones (this strategy is described in greater detail in Chapter 4). The other aim is to recover cohesive grammatically meaningful units in their own right. Inspired by the noun phrase chunking research and problems from the late 1990s (Tjong Kim Sang and Buchholz, 2000), I take the identification of *base noun phrases* (base NPs) as a task manifesting this goal.

More details are given below in how these are implemented in evaluation. To outline, the idea is that a subset of gold-standard tree constituents are extracted and used as a gold standard for evaluation of unsupervised partial parser output. For the first motivation, wherein local constituent predictions are taken as a first step in full parsing, evaluation data consists of *constituent chunks*: non-hierarchical branching constituents extracted from a gold standard constituent tree (Figure 2.6a). In other words, these are the lowest branching nodes. The base NPs extracted from the same tree are illustrated in Figure 2.6b. These are much the same, but do not include the prepositional phrase 'about that'. Neither includes the one word NP 'that', since it is not a multiword constituent. The set of base NPs does not include the NP 'The Cat in the hat', since it contains an embedded NP 'the hat'; base NPs may have some internal constituent structure, but not embedded NPs.

In summary, unsupervised partial parsing is the recovery of useful non-overlapping grammatical constituents. Two direct evaluations are used to characterize *useful* constituents: one based on the structure of treebank tree annotations, and one based on identifying base noun phrases. In this work, "unsupervised chunking" is frequently used to mean the exact same thing as "unsupervised partial parsing".



a) Constituent chunks



b) Base NPs

Figure 2.6: Different subsets of constituents extracted for unsupervised partial parsing evaluation.

2.3.3 Motivating UPP: potential applications

This research focuses on unsupervised partial parsing as a sub-problem of unsupervised parsing, and the primary motivation is to build a better (general) unsupervised parsing system. But, taken on its own, unsupervised partial parsing has the potential to contribute to other human language technologies. Some of these are reviewed here.

Query segmentation In Web search, users often query using multiword phrases which have a special or specific meaning, though often these phrases are mixed into a query with other phrases or individual words. Distinguishing the phrasal units from a 2+ word query is the problem of query segmentation. An example is the difference between a query for "new york times", a phrase probably denoting a single organization, versus "new york times square", where *new york* and *times square* are separate phrases. One simple approach, by Bendersky et al. (2009), makes use of a supervised noun phrase chunker for English. The techniques described in this work could allow a strategy like this to be applied to new languages and domains.

Feature extraction for text classification Finally, UPP has the potential to contribute to a wide variety of text classification problems where word sequences, rather than individual words, may be useful features for classification. An example is *geolocation*, or the classification of texts by a specific point in the world (or within a specific geographical area) associated with the text. Current best approaches for this problem (Eisenstein et al., 2010; Wing and Baldridge, 2011) employ basically the same bag-of-words representation of documents, ignoring potentially useful multiword sequences for this task. In addition to multiword place-names, often multiword phrases are idiomatic of regional dialects, such as "egg cream" (a kind of milkshake, specific to New York City) or "coffee milk" (a popular drink in Rhode Island), sports team names, *etc.*

2.4 Evaluation for unsupervised parsing

As noted, an unsupervised natural language learning task is defined by its evaluation. These sections present the evaluation issues surrounding it for unsupervised parsing and unsupervised chunking.

The standard experimental procedure for unsupervised parsing is as follows: a given system is provided with a text corpus to use as *training data* and another (though possibly the same) text corpus to use as *testing data*. Gold-standard treebank annotations of the test data are used to evaluate the output of the parser – basically, how correct and how complete are the constituents predicted by the parser. However, in unsupervised evaluation, constituent annotations are stripped from the training data (or otherwise ignored, if using text from a treebank).

The evaluation metric used here and elsewhere is unlabeled PARSEVAL. PARSEVAL (Black, 1992) has been the standard evaluation system for supervised parsers for the last two decades. For unsupervised systems, which do not output labeled constituents, the unlabeled variation of PARSEVAL is used. Simply, PARSEVAL measures the precision, recall and *F*-score of complete constituent matches between the predicted structures and the gold standard annotations.

For instance, consider a simple test dataset with just one sentence, and the following actual (gold standard) and predicted annotations:

Gold standard: ((on Sunday) (the brown bear) sleeps)
Predicted: ((on Sunday) (the (brown bear) sleeps))

An individual bracket may be thought of as a pair of indices: the first word index and final word index of the words of the string under the bracket. For instance, in this example, the bracket (on Sunday) would be represented as $\langle 1,2 \rangle$. So, in this notation, the brackets for the example above are:

Gold standard: $\{\langle 1,6 \rangle, \langle 1,2 \rangle, \langle 3,5 \rangle\}$
Predicted: $\{\langle 1,6 \rangle, \langle 1,2 \rangle, \langle 3,6 \rangle, \langle 4,5 \rangle\}$

True positives (TP) are those predictions which match the gold standard annotations, *false positives* (FP) are those predictions which do not correspond to gold standard annotations, and *false negatives* are those gold standard annotations which are not predicted by the system. In this case, the TP = $\{\langle 1,6 \rangle, \langle 1,2 \rangle\}$, FP = $\{\langle 3,6 \rangle, \langle 4,5 \rangle\}$

and FN = $\{\langle 3,5\rangle\}$. *Precision* (Prec) is the ratio of predicted brackets which correspond with gold-standard annotations, or $|\text{TP}|/|\text{TP} \cup \text{FP}|$. In this case:

$$\text{Prec} = \frac{|\{\langle 1,6\rangle, \langle 1,2\rangle\}|}{|\{\langle 1,6\rangle, \langle 1,2\rangle, \langle 3,6\rangle, \langle 4,5\rangle\}|} = 2/4 = .5.$$

Recall (Rec) is the ratio of gold standard brackets which are predicted by the system, or $|\text{TP}|/|\text{TP} \cup \text{FN}|$; in this case:

$$\text{Rec} = \frac{|\{\langle 1,6\rangle, \langle 1,2\rangle\}|}{|\{\langle 1,6\rangle, \langle 1,2\rangle, \langle 3,5\rangle\}|} = 2/3 \approx .667.$$

And *F-score* (or F_1-score) is the harmonic mean of Prec and Rec,

$$F\text{-score} = 2 \cdot \text{Prec} \cdot \text{Rec}/(\text{Prec} + \text{Rec}).$$

In this example, the *F*-score $\approx .571$. The TP, FP and FN for all sentences in the testing dataset are used to calculate Prec, Rec and *F*-score (they are macro-averaged, in other words).

In what follows, Precision, Recall and *F*-score for full constituent identification are reported, multiplied by 100 for readability. Partial overlapping constituents, such as (brown bear) in the example above, simply count as false positives. Also, by these conventions, a system which outputs a bracket covering the whole string gets one true positive for free.

Issues in parser evaluation Beyond the task itself, the resources chosen for evaluation, and the general format of grammatical structure targeted, there are a few other factors affecting the experimental setup and evaluation of unsupervised parsers. As they relate to the experimental results reported later, as well to the overall motivations and aims of unsupervised parsing, these are reviewed in what follows.

2.4.1 Use of POS or word classes

Much foundational research in unsupervised parsing, including the CCM and DMV approaches of Klein and Manning, crucially make use of gold-standard parts of speech during training and evaluation. To be specific, rather than having access to strings of tokens as in

the quick brown fox jumped over the lazy dog,

the systems are trained and evaluated using sequences of POS tags, as in

DT JJ JJ NN VBD IN DT JJ NN

(using Penn Treebank POS tags, see Marcus et al. (1993)). In fairness, supervised approaches to parsing frequently treat POS tagging as a preprocessing step, performed by a special-purpose tool like a hidden Markov model (HMM). An HMM in this setup would itself be based on supervised machine learning techniques, and it is important to note that such a tool only has access to the same training annotations as the full parsing system – a subset, in fact, just the POS annotations.

In the unsupervised case, if the goal of unsupervised parsing is to learn to predict grammatical structure from language with minimal modification – raw text, as articulated in the introduction – then training and evaluation using POS tags in this manner is a significant injection of new information into the parser learning task.

Moreover, assuming gold-standard POS as input to the training procedure, and to the system as evaluation (at run time) raises real questions about the goal of unsupervised parsing research.

H. Daumé III[1] and others raise genuine concerns about research in unsupervised methods: basically, if you, the researcher, have enough annotated material to evaluate an unsupervised parser (for example), then a better solution would be to use this annotated material to train a supervised model (possibly improving such a model with unannotated material in a *semisupervised* manner). Such a model would more than likely outperform an unsupervised parser, and train more quickly.

Such an objection can easily be refuted by unsupervised parsing researchers: while it holds for the languages which are used for direct evaluation, this in no way suggests that an unsupervised parser would not perform comparably for languages or textual domains for which there really is no annotated training material – such as rare languages, or social media like e-mails, weblogs and social networking Websites.

However, the presumption that gold-standard POS are available for a new language or textual domain really does call into question the feasibility of the proposed methods for human language technologies. That is, if you, the practitioner,

[1] `http://nlpers.blogspot.com/2006/04/unsupervised-learning-why.html`

have enough resources to produce high quality POS for a system to use for training and at run time, then indeed you could allocate some resources to produce seed training material for a supervised parser. From another perspective, this all is to say that gold-standard (linguist-annotated) POS, available for training and at runtime, is too big of an assumption; that parsers and parser learning strategies which rely on such an assumption are not really unsupervised, from a practical point of view. Such systems may be unsupervised, in that they are learning novel structural predictions without access to similar representations as training material; but in terms of applicability of the proposed techniques to new languages and domains, and thus the applicability of the proposed techniques to the development of human language technologies, the gold-standard POS assumption is arguably too big a limiting factor.

There is a legitimate response, of course: unsupervised POS models exist, see Christodoulopoulos et al. (2010) for a recent survey. Unsupervised tagging models have been an active area of research over the last few years. Such models are usually evaluated via comparison to gold-standard POS annotations, similar in principle to unsupervised parser evaluation described above, less often are they evaluated as part of a larger system – as in *task-based evaluation*. However, unsupervised parsing is actually a very apt application for unsupervised POS tagging. Unsupervised POS tagging output, like a lot of clustering techniques, can be difficult to apply in a practical setting (for example, in information extraction) since the induced word-classes are essentially unlabeled and difficult to interpret. However, for unsupervised parsing, models like Klein and Manning's CCM neither require nor use any actual understanding of POS tags or word class labels; as opposed to, say, a strategy that requires knowledge of the meaning of POS takes, for instance, identifying noun phrases as determiner-adjective-noun POS sequences. (Such a strategy actually serves as a benchmark in Chapter 3.)

Unfortunately, methods for unsupervised parsing which utilize POS or word class, rather than tokens, frequently perform much worse when using induced word classes. Klein (2005) reports that the performance of CCM when using word classes is nearly seven points less, in *F*-score, than when using gold-standard POS – the performance of this model is only barely beating the right-branching baseline. Headden III et al. (2008) confirm this result, in fact they confirm it with a survey of a wide variety of the best methods for unsupervised POS induction at the time, and

reports a more depressing result: that there does not seem to be any correlation between performance of unsupervised POS taggers – using a number of evaluations against gold-standard tags – and the performance of CCM using induced tags. The same holds for Klein and Manning's (2004) DMV model.

2.4.2 Annotation conventions

Comparison against human annotated treebanks is the best option for direct evaluation of unsupervised parsers, however it is hardly a perfect one. All empirical evaluations of natural language processing technology have problems, but unsupervised parser evaluation is particularly problematic due to our (imperfect) scientific understanding of human language syntax, and the (imperfect) encoding of that understanding in treebanks.

First of all, treebank evaluation consists of comparing results of a computer program to linguists' annotations, which can be inconsistent. Many evaluations in natural language processing make use of human annotations in this way (part of speech tagging, for example), though others are not. For instance, text regression (Joshi et al., 2010) is the task of predicting real world numerical quantities (*e.g.* stock prices or movie ratings) based on text. However, human language grammatical structure is far more subtle. There is even a lack of consensus as to whether grammatical structure is best characterized by constituency trees – which are used in this work, and go as far back as Chomsky (1957) or even possibly Bloomfield (1933) – or dependency trees (Tesnière, 1959; Mel'čuk, 1988), or category-driven approaches wherein syntactic processing is more akin to logical derivation (Moortgat, 1997; Steedman, 2001).

Even accepting constituency trees, in the abstract, as a good-enough representation of grammatical structure (or some level thereof), it is not the case the different treebank datasets adhere to the same conventions or standards in their annotations. Most large-scale natural language annotation projects are careful to design, publish, and adhere to a set of *annotation conventions* which dictate the theoretical basis for the choices made, as well as specific guidelines for annotators. All the treebanks considered in this thesis certainly do. The conventions articulated by a set of annotation conventions can differ significantly from one resource to the next, even within the same language.

Two of the resources used as primary resources in this research, the Penn Treebank and the Penn Chinese Treebank, are the subject of a case study by Levy and Manning (2003). In addition to genuine linguistic differences between English and Chinese, these treebanks differ according to the underlying linguistic theory guiding the annotation effort. Whereas the (English) Penn Treebank annotations were influenced by 1970s Transformational Grammar, the Chinese Treebank were influenced by the 1980s Government and Binding (GB) grammar tradition (*e.g.* Chomsky, 1986). In the later, specific constituent structure patterns are used to characterize different verb-modifier relationships – specifically adjunction and complementation – leading to overall more constituent structure with a lower branching factor, compared to the Penn Treebank's relatively flat trees. Another feature of the GB grammar tradition is that lexical categories are standardly associated with phrasal category projections – sub-trees specifically associated with lexical category – which, for the purposes of unsupervised parser evaluation using unlabeled trees, also leads to overall more constituent structure and a lower branching factor than the Penn Treebank.

The other resource used during model development is the Negra German corpus, and it differs from CTB and PTB in a much more drastic way (Krenn et al., 1998). The conventions developed for Negra annotation were intended for *free word-order* languages in general. More specifically, languages which (like German) will have some phrasal structural effects, best characterized by constituency annotations, but other effects – like local scrambling – are better characterized by word-to-word (bi-lexical) dependencies. For this reason, Negra consists of *two levels* of treebank annotations: both dependency annotations and constituency annotations.

However, the methods proposed here are only evaluated using the constituency annotations, as in other research on unsupervised parsing using Negra (Klein and Manning, 2004; Seginer, 2007a; Hänig, 2010). Dependency annotations are ignored in these evaluations, in other words. This means that some of the legitimate grammatical structure that is present in the language – and may be identified by parsing systems – is not present in the evaluation material used, even if it is present in the separate dependency annotations. An example of this is the annotation of noun phrases (NPs) within prepositional phrases (PPs). While NPs are annotated as constituents elsewhere, PPs are treated as flat constituent structures, where the relationship between the preposition and the subordinate phrasal terms

	WSJ	Negra	CTB
words per sentence	21.4	15.1	21.7
constituent length	8.1	6.6	7.0
constituent length S20	7.3	6.5	6.0
constituents per sentence	14.5	6.8	17.2
constituents per sentence normalized	14.2	9.4	15.4

Table 2.1: Corpus statistics, including average sentence length, average constituent length, average constituent length for sentences containing exactly 20 words (S20), the average number of constituents per sentence and the normalized average number of constituents per sentence. The normalized average constituents per sentence is the average number of constituents per word x 20.

is only indicated in the dependency annotations. Examples include "auf die Wiesbadener Staatsanwaelte" (on Wiesbaden's district attorneys) and "in Hannovers Nachbarstadt" (in Hannover's neighbor city).

These annotation conventions have quantifiable effects on the nature of the treebank dataset. Basically, with its emphasis on binary-branching trees, the Chinese Treebank favors relatively short constituents compared to WSJ – see Table 2.1. The Negra treebank, it turns out, does not have quantifiably longer constituents, but it does contain far fewer constituents per sentence – even when normalized for sentence length. The effect is that Negra trees are flatter than WSJ or CTB.

The lesson here is that the best-performing models, in terms of direct evaluation using treebanks, are in part those which "learn" the annotation conventions as well as the grammatical structure of the language itself. Different model classes contain basic biases in terms of the kinds of structures targeted; sometimes there is a good match between model biases and annotation conventions, other times a poor match. For this reason, it is important to evaluate parsing methods on multiple treebanks employing different methologies.

2.4.3 Phrasal punctuation

Whether POS tags are used or raw text, standard unsupervised parser evaluation does not count punctuation in evaluation. That is to say, the output of unsupervised parsers is a bracket structure over a string of terms (words or tags), with no punctuation output; this structure is evaluated by comparison to brackets taken from gold-standard treebank annotations, again with punctuation removed. Com-

pare Figure 2.3 and Figure 2.4 on 12 for an example. In supervised parser evaluation, too, punctuation is basically treated in an ad hoc fashion as well (Hollingshead et al., 2005; Collins, 1999).

In much previous research, punctuation is ignored in training and evaluation material, as well. That is, for a model like Klein and Manning's CCM, input to the model for training and evaluation would simply be a string of POS tags, wholly omitting tags or any other indicators for punctuation. This holds for constituency models, like those of Bod (2006a,b), contemporary with CCM, and more recent work in dependency models, stemming from Klein and Manning (2004)'s DMV (Smith and Eisner, 2006; Headden III et al., 2009; Cohen and Smith, 2009; Spitkovsky et al., 2010a). Seginer (2007a) departs from this tradition by making use of what is here called *phrasal punctuation* – punctuation symbols that often mark phrasal boundaries within a sentence. The set of punctuation used by Seginer and in this thesis research is

. ? ! ; , -- 。 、

The latter two are Chinese full-stop and ideographic comma.[2] Other punctuation – parentheses, quotes, *etc.* – are dropped from the input (training and evaluation) from the system. These symbols are not, note, treated simply as terms like any others. These are used by Seginer's CCL parser in two ways: i) they impose a hard constraint on constituent spans, in that no constituent (other than sentence root) may extend over a punctuation symbol, and ii) they contribute to the model, specifically in terms of the statistics of words seen adjacent to a phrasal boundary.[3]

CCL, then, uses a pre-processing step to identify punctuation, and keep a subset of it to constrain the model. Of course, the other research mentioned above uses a similar preprocessing step simply to identify punctuation, before it is removed from the dataset. That said, all of the systems discussed in this dissertation make use of sentence boundaries.

In Chapter 4, both CCL and CCM are evaluated with and without phrasal punctuation, as are the new methods proposed in this thesis. This is a simple but novel extension to CCM, enabling a direct comparison of this sort for the first time.

[2]While it is clear from our analysis of CCL that it does make use of phrasal punctuation in Chinese, I am not certain whether ideographic comma is included in the set used.

[3]In addition to Seginer, in recent work Spitkovsky et al. (2011) present research on incorporating punctuation and constraints related to it to constraining an unsupervised dependency parser.

2.4.4 Sentence length

Klein and Manning (2002, 2004) established a common practice in the field of unsupervised parsing research, wherein only sentences of length $\leq$10 are used for training and evaluation. This technique not only enables experiments to be run in reasonable time (since CCM operates in polynomial time as a function of sentence length, it performs much slower on long sentences than on short), but also adequately constrains the learning algorithm to learn to predict grammatical constituent structures.

What is at work is an analogue of McClosky et al. (2006)'s "Goldilocks effect" in a purely unsupervised environment: McClosky et al. investigate using unlabeled sentence data to improve a supervised parser, and find that while trivially (one or two word) sentences were uninformative for their purposes, long sentences were also unhelpful. Model reestimation tended to overfit when given long sentences for training.

From the perspective of developing parser learning methods for human language technology, evaluation only using short sentences is problematic, since most sentences in textual prose are longer – the average sentence length in WSJ is 20.9 tokens, in Negra it is 14.7, in CTB 23.1. In fairness, recent research in unsupervised dependency parsing actually *evaluate* using full-length sentences (Headden III et al., 2009; Cohen and Smith, 2009; Spitkovsky et al., 2010a,b,c), even though the training set consists of shorter sentences.

Spitkovsky et al. (2010a) explores this in unsupervised dependency parsing, using Klein and Manning (2004)'s DMV. He finds that the "sweet spot" – the best length of sentences for training DMV – is actually 15. Moreover, a cascaded technique which starts by training a model with short (length = 1) sentences, and incrementally refines the model with longer sentences, peforms better still.

Seginer (2007a)'s CCL parser works better on short sentences, but does not degrade as dramatically with longer sentences. Of course, CCL has the benefit of using phrasal punctuation, which has the effect of partitioning long strings into a set of shorter sub-sequences.

2.4.5 Held-out evaluation

Finally, much previous research in unsupervised parsing essentially trains and evaluates using the same dataset. Take the CCM, for instance, of Klein and Manning (2002): the standard experimental setup is that the system is provided with the full set of length $\leq$10 sentences – POS tags, punctuation omitted – and the system outputs a set of bracketing structures over these.

Again, from the perspective of developing human language technologies, this is an unrealistic experimental scenario. Most unsupervised parser learning techniques rely on batch-learning, thus would have to be completely retrained in the event a new sentence is encountered in some text-mining application. The CCL parser is a notable exception, though in practice the code-base supporting the research only implements a batch-learning setup. Moreover, all of the learning techniques reviewed here learn a general parser (including CCL), which can be evaluated on held-out evaluation material. Also all techniques reviewed here are much faster at parsing using a trained model, than at actually training a model (including CCL: in spite of the fact that it trains very fast, it parses with a trained model much faster).

Recent research in unsupervised dependency parsing has moved to use of separate training and evaluation datasets (work by Headden III et al., Cohen and Smith, and Spitkovsky et al. cited above). For evaluation using WSJ, the test dataset is the same – section 23 – as is standardly used in supervised parser evaluation.

In addition to training and held-out evaluation datasets, a *development dataset* is used to experimentally determine good parameters and strategies; a *blind test* dataset is not used in experimentation until final evaluation of the proposed methods is conducted. Unless otherwise noted, the experimental numbers reported in this work are based on a blind test dataset.

2.5 Evaluation for unsupervised partial parsing

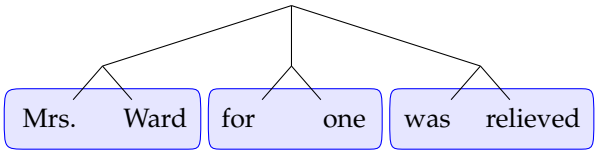
One of the contributions of this thesis research is to establish a new problem in unsupervised grammar learning: *unsupervised partial parsing*, or simply *unsupervised chunking*. This task is similar to an unsupervised analog of the supervised chunking and noun phrase identification tasks, (Ramshaw and Marcus, 1995); also, this

is similar to an analog of named-entity recognition (NER) (Tjong Kim Sang and De Meulder, 2003). In each, the evaluation is based on the ability of models to identify and label sequences of terms which correspond to some specified unit of representation: for noun phrase (NP) identification, the target is the identification of NPs; for chunking, the target is NPs and other sequences of words, as well as the phrasal type; for NER, the target is the identification of named-entities – different kinds of NE include person, place and organization – and a label corresponding to which kind of entity. In each of these tasks, the sequences of terms identified must not be overlapping; their length may be one word or more.

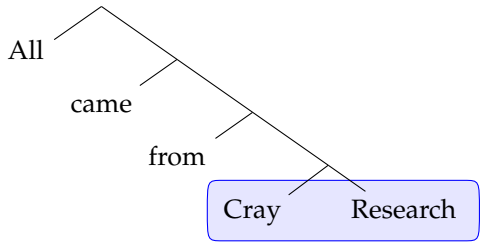
As supervised NLP tasks, models have access to training material, which basically defines the task for the model, and provides the labels for phrasal type or kind of named entity. For the unsupervised case, here, a given model only has access to strings of terms, so like unsupervised parsing evaluation is only on the identification of relevant non-overlapping constituents, not their labels. Also, single-word constituents are regarded as indistinguishable from individual words *not* wrapped in a constituent, so only *multiword* constituents are to be considered.

An unsupervised chunker predicts unlabeled constituents, like unsupervised constituent parsers, but only non-overlapping constituents. Because of the connection to full constituency parsing, which is explored in Chapter 4, a subset of treebank annotations is used as a gold standard for direct evaluation of unsupervised parsing techniques (the same treebanks which are used elsewhere in this work). The constituents must be non-overlapping, of course, but also represent a meaningful view of low-level grammatical structure.

In fact, three evaluations are proposed here for unsupervised chunker evaluation, representing slightly different takes on low-level constituent structure. Firstly, **Constituent chunks** (often simply refereed to as *chunks* in what follows) are the subset of gold standard constituents which are i) branching (multiword) but ii) non-hierarchical (do not contain subconstituents). Secondly, **base noun phrases** (base NPs), are the subset of NPs that do not contain nested NPs; where base NPs have internal structure other than nested NPs, they are basically flattened and treated as a single constituent. Examples of constituent chunks extracted from treebank constituent trees are in Figure 2.7. Evaluation of unsupervised chunking systems is basically unlabeled PARSEVAL applied to these constituents – precision, recall and *F*-score on identification of chunks and NPs. Additionally, chunking



(a) Chunks: (Mrs. Ward), (for one), and (was relieved)



(b) Only one chunk extracted: (Cray Research)

Figure 2.7: Examples of constituent chunks extracted from syntactic trees

predictions can be evaluated by comparison to full treebank annotations, as in the *precision* of chunking predictions against all annotated constituents. This is the same as evaluating chunking constituents by the proportion which do not cross boundaries with gold standard annotated constituents. Note that this means that different evaluations are run on the same output of an unsupervised chunker.

Direct evaluation versus task-based evaluation In §2.3.3, I cited applications of unsupervised chunking to other tasks in human language technologies, beyond unsupervised parsing. *Task-based evaluation* of UPP models would establish how much and in what ways a current approach to a the task can be improved by using unsupervised chunking. The evaluation used in this dissertation, however, is direct comparison to treebank annotations. This is for a couple of reasons.

The primary view of unsupvised chunking, in this dissertation, is that it is a sub-task of general unsupervised parsing. Thus, I adopt evaluations for unsupervised chunking that relate this task to unsupervised parsing evaluation, using subsets of the standard treebank annotations used to evaluate unsupervised parsers. And, I adopt the standard treebank annotations to compare the output of the unsupervised chunking systems, and cascaded parsers build from unsupervised chunkers, to parsing methods established in the field, CCL (§2.6.2) and CCM (§2.6.1) in particular.

Yet, this reliance on direct evaluation has its limitations. Shownig that an unsupervised parser or unsupervised partial parser improves on the state-of-the-art in direct evaluation does not guarantee it will contibute to improvements in task-based evaluation. Moreover, human syntactic annotations are imperfect, and different treebanks have different conventions, with tangible effects on the kinds of structures annotated (§2.4.2), though the parsers evaluated obviously do not have access to these distinctions.

2.6 Background and benchmarks

Good surveys of the history of unsupervised grammar induction – at least until 2005 and 2007 – are given by Klein (2005) and Seginer (2007b); I will summarize this history briefly, but see these surveys and works cited therein for more details.

Broadly speaking, there are a few proposed solutions to the problem of learning to parse natural language in an unsupervised fashion. In one, alluded to above, the parameters of a probabilistic context free grammar are estimated, using an expectation maximization (EM) method like the inside-outside algorithm (Lari and Young, 1990). Unfortunately, these methods are problematic: they have trouble recovering artificial grammars (Lari and Young, 1990) and, given different initializations, fail to converge to consistent grammars, and do not make accurate predictions (Carroll and Charniak, 1993). One more positive result from this effort is that of Pereira and Schabes (1992), who show that the inside-outside can learn a fairly good grammar, with the strong precondition that bracketings are available for the training material. In other words, this result showed that a) the inside-outside algorithm could learn a PCFG which effectively *labels* unlabeled constituent trees, and b) the resulting grammar could be deployed on new text with reasonable success. Klein (2005) reports an alternative approach to learning context-free grammars from GPOS data, Greedy-Merge, and while the grammars learned by this system have some linguistic validity, they too fail to beat a right-branching baseline when evaluated empirically.

A variety of approaches – such as EMILE (Adriaans et al., 2000), ABL (van Zaanen, 2000), and ADIOS (Solan et al., 2005) – basically follow a similar strategy. They work by trying to identify sequences of word types which can appear in different contexts: that is, sequences that can be substituted for each other, and thus

are likely of the same constituent type. As such, these efforts basically attempt to implement algorithmically some of the intuitions of the constituency tests, mentioned above, as a means of learning constituents. Unfortunately, these methods are often inefficient to run on large corpora, and when evaluated empirically do not beat simple baselines (Cramer, 2007). Brooks (2006) also presents a variation on this strategy, which has deep connections to the methods proposed here (though developed independently) – see Chapter 4.4 (p. 115) for a discussion.

Finally, there are models which learn a probability distribution directly over the grammatical structures under consideration. These include the constituent context model (CCM) of Klein and Manning (2002), which is described in greater detail below; a variation of CCM by Smith and Eisner (2004) which uses deterministic annealing rather than expectation maximization; the dependency model with valence (DMV) of Klein and Manning (2004), an unsupervised generative model for dependencies (rather than constituents); and a succession of dependency models as variations on DMV, including those of Cohen et al. (2009); Cohen and Smith (2009); Headden III et al. (2009) and Spitkovsky et al. (2010a,b,c, 2011). Bod (2006a,b) presents models which learn probabilistic data-oriented parsers (DOP) from unlabeled material by simulated training using all possible trees over the dataset. Finally, Blunsom and Cohn (2010) learn a probabilistic dependency parser as a tree-substitution grammar.

2.6.1 The generative constituent-context model (CCM)

As I will come back to this model in what follows, I present some of the details of the CCM model here. This is the simplest of the recent crop of probabilistic unsupervised models, but it is one of the only ones to target constituency structure directly. At its core, CCM is a generative model over bracketings b over a string of symbols (a sentence) $\mathbf{x}$.[4] What is defined is a joint probability distribution $\Pr(\mathbf{x}, b)$ such that, given a sentence $\mathbf{x}$, this distribution is used to choose a bracketing by identifying the most likely: $\operatorname{argmax}_b \Pr(\mathbf{x}, b)$. The joint probability is factored into different parts in the usual way:

$$\Pr(\mathbf{x}, b) = \Pr(b) \Pr(\mathbf{x}|b)$$

[4] A discussion of related generative probabilistic models is in §3.2.1 (p. 47).

where $\Pr(b)$ – the prior probability of a bracketing – is uniform over bracketings which correspond to binary trees; the conditional probability $\Pr(\mathbf{x}|b)$ is the likelihood of a sentence $\mathbf{x}$ given a bracketing b. This is calculated with respect to parameters c and C – for constituent likelihoods – and d and D for distituent likelihoods. c and d define probabilities over strings: $c_{x_{i...j}}$ is a probability that $x_{i...j}$ is a constituent, where $x_{i...j}$ is a substring of a string $\mathbf{x}$; $d_{x_{i...j}}$ is a probability that the sub-string $x_{i...j}$ is a *distituent* (*i.e.* not a constituent). The *context* probabilities C and D are defined over contexts, defined as pairs of words of a string; so, $C_{\langle x_i, x_j \rangle}$ is a probability that the words x_i and x_j surround a constituent, likewise $D_{\langle x_i, x_j \rangle}$ is a probability that x_i and x_j surround a distituent, *i.e.* that $x_{i+1,...,j-1}$ is not a constituent. In context probabilities, x_0 and $x_{|\mathbf{x}|+1}$ are set to special sentence boundary symbols. A CCM model consists of a fixed set $\theta = (c, d, C, D)$ understood this way. A bracketing b is taken to be set of pairs of indices (i, j) where $i < j$; for convenience, say $b_{ij} = 1$ if $(i, j) \in b$, $b_{ij} = 0$ otherwise. So, now, given a bracketing b and a CCM model $\theta = (c, d, C, D)$, the probability of a sentence is given as

$$\Pr(\mathbf{x}|b; c, d, C, D) = \prod_{i=1}^{|\mathbf{x}|-1} \prod_{j=i+1}^{|\mathbf{x}|} b_{ij} c_{x_{i...j}} C_{\langle x_{i-1}, x_{j+1} \rangle} + (1 - b_{ij}) d_{x_{i...j}} D_{\langle x_{i-1}, x_{j+1} \rangle}$$

Since $\Pr(b) = 0$ for non-binary trees, the estimation of $\Pr(\mathbf{x}|b; c, d, C, D)$ can assume b is a binary tree, which moreover means that this probability may be calculated efficiently via dynamic programming using a data-structure similar to a CYK table (*c.f.* Manning and Schütze, 1999; Jurafsky and Martin, 2008, for CYK tables). A CCM model θ is learned by estimating the most likely model given a set of sentences $\mathcal{X}$:

$$\hat{\theta} = \operatorname*{argmax}_{\theta} \sum_{\mathbf{x} \in \mathcal{X}} \sum_{b} \Pr(\mathbf{x}, b; \theta)$$

This cannot be estimated directly, so is estimated via expectation-maximization (EM), an iterative hill-climbing procedure for optimizing $\hat{\theta}$. EM is discussed in the context of hidden Markov models in §3.2.1. For *parsing*, since b is guaranteed to be a binary tree, a Viterbi algorithm akin to CYK parsing is available to calculated $\operatorname{argmax}_b \Pr(\mathbf{x}, b; \theta)$. This outline paints CCM in general strokes, see Klein and Manning (2002) and Klein (2005) for details.

2.6.2 The common cover links parser (CCL)

One other system merits extended discussion, as background to this work: the common cover links parser (CCL parser) of Seginer (2007a,b). CCL is a notable innovation in unsupervised parsing research, in that it is effective at raw text unsupervised parsing, without requiring POS or a preprocessing step learning word classes. Also: it is very efficient in space and time, during training as well as at runtime. CCL makes use of a novel representation of constrained constituent syntactic structure, which limits the search space of possible parses. On the other hand, it does not assume binary branching trees. However, the learning and parsing methods are difficult to articulate, understand, replicate or extend. This means that direct improvement is difficult. New research *has* improved on CCL's results, but by selecting training sentences specifically for the evaluation sentences (Reichart and Rappoport, 2010).

Common cover links are binary relations over tokens in a sentence which uniquely identify constituent structures (i.e. bracketing structures or syntax trees). The basic idea is that if there is a link between word n and word m with depth d (denoted $n \xrightarrow{d} m$) then $n \neq m$, n and m are under the same node K in the tree, and n is the shallowest token under K, or of equal depth as m; the depth d is the number of non-terminal nodes between K and n.

Consider the syntax tree in Figure 2.8. Since the first "the" token is as high or higher in the tree than any other tokens, then among the possible links characterizing this tree are

$$\text{the}_1 \xrightarrow{0} \text{cat} \qquad \text{the}_1 \xrightarrow{1} \text{red}$$
$$\text{the}_1 \xrightarrow{1} \text{saw} \qquad \text{the}_1 \xrightarrow{1} \text{dog}$$
$$\text{the}_1 \xrightarrow{1} \text{the}_2 \qquad \text{the}_1 \xrightarrow{1} \text{run}$$

Shortest common cover link sets are sets of CCL which unambiguously denote a single tree, but contain a minimum number of CCL. One characterization of shortest CCL sets is minimality with respect to transitivity of CCL; in the example above, e.g., given that there are links $\text{cat} \xrightarrow{1} \text{saw}$ and $\text{saw} \xrightarrow{0} \text{the}_2$, the link $\text{cat} \xrightarrow{1} \text{the}_2$ is redundant. A shortest CCL set corresponding to the tree in Figure 2.8 can be illustrated perspicuously as in Figure 2.9.

The task of parsing is to map a sequence of tokens to a bracketing represen-

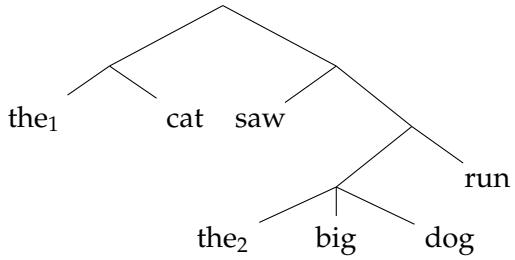


Figure 2.8: Example syntax tree (tokens of the same word type are distinguished with indices)

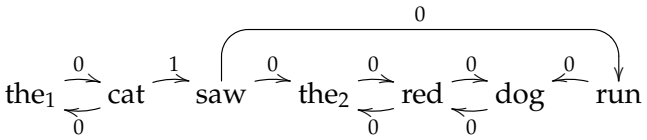


Figure 2.9: Example shortest CCL set

tation (*i.e.* a tree). In CCL parsing, this is recast as mapping a sequence of tokens to a shortest CCL set, from which a bracketing can be straightforwardly derived. In short, CCL parsing consists of iteratively choosing one from a set of addable links given a (partial) CCL set L and an index $2 \leq k$ in the sentence. The available links to choose from are provided by the formalism: a link may only be added at an unused adjacency including x_k, may only be added to the left of x_k and must adhere to additional constraints guaranteeing the link set corresponds to a tree. *Which link* to choose is determined by a weight function. The weight function, in turn, is based on a CCL lexicon, which is bootstrapped from a corpus.

Learning a CCL parser corresponds to learning a lexicon, which store a variety of statistics for each word type in the corpus. These statistics are associated with *adjacency positions* $R1$, $R2$, $L1$, etc. corresponding to how frequently a word type is seen with other word types (and sentence boundaries, etc) to the right or the left, and at what distance in terms of adjacencies. For example, in Figure 2.9, at the $R1$ position from 'saw' is 'the$_2$'; at the $R2$ position from 'saw' is 'run'; and at the $R3$ position is the sentence boundary. A lexicon is bootstrapped from a corpus in the following way: a parser is initialized with the empty lexicon, and parses the first sentence of the corpus; the lexicon is updated with counts from the parser output, and the (updated) parser parses the second sentence; and so on.

2.6.3 Benchmarks

Heretofore, in the unsupervised constituent parsing problem space, no single system has been shown to adequately satisfy all of the desiderata listed in the introduction (p. 4): 1. operate on raw text, 2. extensible, 3. efficient and 4. state of the art performance under empirical evaluation. Of the prior work, two of the models outlined above are used here as benchmarks for comparison: the constituent context model (CCM) of Klein and Manning (2002), and the common cover links (CCL) parser of Seginer (2007a). Both basically satisfy desiderata 3 (efficiency) and 4 (state-of-the-art performance), but represent trade-offs in terms of desiderata 1 and 2.

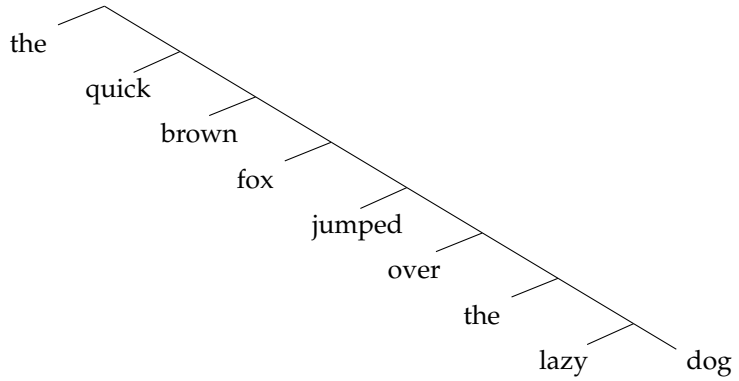
CCM defines a probability distribution over bracketings, and has a learning strategy for the parameters of the probability distribution using EM. This simple and straightforward approach is certainly extensible, and in fact other work has built on this foundation (Smith and Eisner, 2004; Snyder et al., 2009). For short sentences, the simple strategy is efficient to train and run – though due to its cubic time algorithm, it is much less efficient for longer sentences. Its empirical performance degrades for longer sentences, too. And, CCM makes crucial use of hand-annotated (gold-standard) POS sequences for training and evaluation, and does not operate effectively on raw text.

CCL is trained and evaluated making direct use of raw text, without the usage of gold-standard POS or word-classes. It is notably efficient for training and evaluation (or, at run-time). However, the underlying representation of syntactic structure – common cover links – and, moreover, the parser learner and parsing algorithms are complex, rule-based, heuristic approaches which are difficult to replicate, and harder still to extend and improve on.

2.7 Baselines

In addition to benchmarks, representing alternative predictive approaches, it is important to establish non-trivial baseline results as a basis for comparison. These baselines are simple, basic heuristics for generating trees (for unsupervised parsing) and chunks (for unsupervised chunking) that nevertheless may capture some basic insight about the natural language grammatical patterns being explored. They

Input: the quick brown fox jumps over the lazy dog



(the (quick (brown (fox (jumped (over (the (lazy dog))))))))

Figure 2.10: Simple right-branching baseline parse (tree and brackets).

should, nevertheless, be considered the lower bound of acceptable performance for approaches to these problems.

2.7.1 Unsupervised parsing baselines

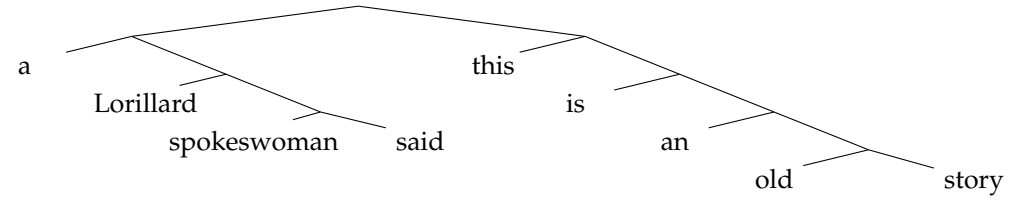
Heuristic baselines The standard baseline for unsupervised parsing, at least of English, is the *right-branching baseline*. That is, over a string of words, posit a simple tree structure that only branches to the right, as in Figure 2.10. While very simple, for languages like English (and annotation guidelines like those of the Penn Treebank), right-branching baseline parses such as these can often capture a reasonable number of correct predictions. In this example, the full VP

(jumped (over (the (lazy dog))))

is basically correct.

Seginer (2007b) introduces a stronger variation of these baselines. Since the CCL model makes specific, explicit use of phrasal punctuation to constrain parser output, Seginer uses a baseline which does as well. Basically, the heuristic is to output a right-branching bracket structure over every sequence of terms bounded by phrasal punctuation; these trees are then put under a common root node. An example of this is given in Figure 2.11. In this example, the phrasal punctuation component of the heuristic contributes to an additional true positive bracket:

Input: a Lorillard spokeswoman said , this is an old story .



((a (Lorillard (spokeswoman (said)))) (this (is (an (old (story))))))

Figure 2.11: Punctuation sensitive right-branching baseline parse (tree and brackets).

(a Lorillard spokeswoman said)

In what follows, the punctuation specific right-branching baseline (RB) is used.

Random baselines In addition, a random-tree baseline is evaluated. Random binary trees are constructed by choosing to boundaries in a bracketing structure randomly. For consistency with the other baselines, again the first branching node basically spits the tree into subtrees by phrasal punctuation; each subtree is a randomly constructed binary branching tree.

2.7.2 Chunking baselines

As with parsing, baselines for unsupervised chunking are established by considering the output and evaluation of alternative – usually, simpler – means of identifying constituents.

Chunks extracted from unsupervised parsers One of the theses of this work is that low-level constituents can be more accurately predicted by models targeting these grammatical structures directly rather than by general unsupervised parsers. For this reason, lowest-level constituent predictions of the CCL unsupervised parser are used as a baseline[5] for comparison. That is, the same rules described above for identifying constituent chunks from gold standard parses are applied to the parser

[5]Since this is more than a trivially simple means of constructing constituent chunk predictions, I often refer to these as *benchmarks* rather than baselines.

output. These usually have higher precision than the other constituents identified by the parser, and they seem to predict low-level constituents comparatively well.

This not a wholly unrealistic usage of CCL: Ponvert et al. (2010) show that a) this representation of non-hierarchical multiword constituents is implicit in the common cover links representation of constituent structure, and b) while the CCL parser usually outperforms a right branching baseline in empirical evaluation, this turns out to be based on the relatively high precision of precisely these low-level constituents. Otherwise, CCL output tends to be relatively flat, forests of relatively high precision local constituent predictions. When CCL's local constituent predictions – the constituents used as a benchmark here – are combined with a right-branching baseline, the resulting constituent output are actually closer to gold standard than the output of the parser itself in English (WSJ) and Chinese (CTB).

That said, the author of the CCL parser never necessarily intended it to be used in this way, nor for its local predictions to be extracted from the parser output and evaluated as such. For this reason, two other baselines are used for comparison.

Treebank constituent precision Local constituent predictions are likely to have higher precision when evaluated on treebank annotations than a constituent parser's overall predictions. For this reason, predictions output by the unsupervised chunking models are subjected to a kind of head-to-head evaluation with CCL parser, looking specifically at the precision of these predictions. Since the chunking models are only predicting one layer of constituent structure, they will usually have much lower recall than CCL, so only precision is considered.

Chunks extracted from POS sequences One of the evaluations for unsupervised chunking is base NP identification; another, constituent chunks, overlaps with NPs in the datasets under consideration. For this reason, a baseline for chunking/base NP identification is constructed by identifying sequences of gold standard POS tags corresponding to noun phrases. For instance, determiner-adjective-noun noun phrases can be identified by this pattern of (Penn Treebank) POS tags:[6]

```
DT JJ* NN|NNS
```

[6]For the Penn Treebanks tagset, see Marcus et al. (1993). The tags used here are `DT` = determiner, `JJ` = adjective, `NN` = common noun, `NNS` = plural common noun.

Similarly, proper nouns can be relatively well predicted by the POS sequence

```
(NNPS?)+
```

This can be generalized, given training data. The idea is: if you had POS for training data and for test data and knew, from the training data, which POS sequences corresponded to the brackets under consideration – constituent chunks or base NPs – then how well could you do by matching known sequences of POS? Take the task of base NP identification: the first step is to compile the N most frequent POS sequences corresponding to NPs in the training data; then, in the test data, any POS sequence matching one of the known NP POS sequences is identified as an NP. This is done by greedy matching, so the longest POS sequence matching a known NP is selected. This strategy involves the parameter N, and so two benchmarks are actually used: GPOS-5, which fixes N at 5, and GPOS-$\star$, which involves a very good guess at N; specifically, the best performing GPOS benchmark for $N \in \{5, 10, 25, 50, 100, 250, 500\}$.

Supervised benchmark Finally, the proposed models are compared to a supervised benchmark, which is learned from treebank annotations for constituent chunks or base NPs from the training material. The details are based on the design of the proposed models themselves, and so I present these benchmarks in the following chapter.

2.8 Datasets

Direct evaluation of parsing systems is done by comparison of parser output to a *gold-standard* consisting of annotations by linguists on language data in context. Datasets of this sort – *treebanks* – have been crucial to the development of natural language parsing techniques, both in providing training material for supervised approaches, and as evaluation material for both supervised and unsupervised approaches.

Because this thesis focuses on unsupervised methods, treebank annotations are used to evaluate the methods proposed. The methods proposed in this work – as well as alternative methods used as benchmarks for comparison – are evaluated on English, German and Chinese data, using resources which have become

standards for unsupervised parsing evaluation. All three are corpora of news text, thus the textual domain is roughly equivalent; they span three languages from two language families.

For English, I use the Wall Street Journal subset of the Penn Treebank-3 (**WSJ**) (Marcus et al., 1999);[7] this is a news text corpus featuring articles from the Wall Street Journal from 1989. The sections used in this research (which is most of them) contain almost 44 thousand (44K) sentences and around 1 million (1M) words, and remains one of the largest and most carefully engineered resources of its kind.

For German, the Negra corpus v2 (Krenn et al., 1998) is used for evaluation.[8] This is a corpus of German newspaper text from the *Frankfurter Rundschau*; the text is also part of the ECI Multilingual Text corpus.[9] In total, there are 20,602 sentences with about 342K words.

For Chinese, the Penn Chinese Treebank v5.0 (**CTB**) (Palmer et al., 2005).[10] This corpus consists of 18,612 sentences with 427K words of Chinese newspaper and newswire text from Xinhua (1994–1998), the Information Services Department of HKSAR (1997) and Sinorama magazine, Taiwan (1996–1998 & 2000–2001). A feature of the later versions of this resource (v4 and above) is the inclusion of a number of new files from HKSAR and Sinorama, 187 of the total 885 documents. Moreover, the textual domain of these new sources is fairly different from the original sources; this impacts the choice of training data-testing data split.

Many factors go into the design of a treebank, which have a tangible effect on the evaluation of supervised and unsupervised methods which utilized these resources for evaluation. Some of these are reviewed in §2.4 below.

Data for unsupervised chunking evaluation For evaluation data, constituent chunks and base NPs were extracted from each of the treebanks described above. In English (WSJ) treebank annotations, constituent chunks largely correspond with base NPs, but this is less the case with Chinese and German. Moreover, the relationship between NPs and constituent chunks is not a subset relation: some base NPs do have internal constituent structure. The numbers of constituent chunks

[7] `http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC99T42`
[8] `http://www.coli.uni-saarland.de/projects/sfb378/negra-corpus/`
[9] `http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC94T5`
[10] `http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2005T01`

WSJ	Chunks	203K
	NPs	172K
	Chunks ∩ NPs	161K
Negra	Chunks	59K
	NPs	33K
	Chunks ∩ NPs	23K
CTB	Chunks	92K
	NPs	56K
	Chunks ∩ NPs	43K

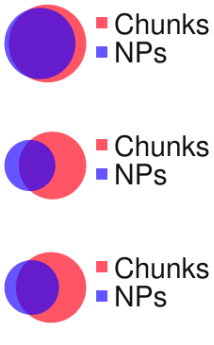


Table 2.2: Constituent chunks and base NPs in the datasets.

		% constituents	% words
WSJ	Chunks	32.9	57.7
	NPs	27.9	53.1
Negra	Chunks	45.4	53.6
	NPs	25.5	42.4
CTB	Chunks	32.5	55.4
	NPs	19.8	42.9

Table 2.3: Percentage of gold standard constituents and words under constituent chunks and base NPs.

and NPs for the training datasets are in Table 2.2. The percentage of constituents in these datasets which fall under these definitions, and the percentage of words under these constituents, are in Table 2.3.

For WSJ (English) sections 00-22 are used for training, section 23 for a blind test set and sections 00-01 are used for development; for Negra (German) the first 18,602 sentences are used for a training dataset, the last 1000 sentences for development and the penultimate 1000 sentences for a blind test dataset; for CTB (Chinese) I adopt the training/development/blind test data-split of Duan et al. (2007). This is due to the fact that contiguous regions of the CTB (*i.e.* sequential files) tend to be clumped by source and domain, thus remixing the source of the training and evaluation provides a clearer view of model performance. This is true of previous versions of CTB (Levy and Manning, 2003), and more so with new data added in later versions (Duan et al., 2007).

	Sentences	Tokens	Types	Hapax	% Types	% Tokens
WSJ	41,532	867,811	40,280	18,177	45.1	2.1
Negra	18,602	274,036	45,616	28,359	62.1	10.3
CTB	15,904	279,124	34,401	16,975	49.3	6.1

Table 2.4: Basic corpus statistics for WSJ, Negra and CTB train data-sets.

Corpus statistics Table 2.4 summarizes basic corpus statistics for WSJ, Negra and CTB. These numbers report statistics for the training subsets, since this represents the text used to estimate model parameters in the models described in subsequent chapters. Also: to provide one sense for the textual differences between the corpora, statistics for *hapax legomena* – terms which occur only once in the corpus – are given. These statistics given some sense for the challenges posed by these corpora, Negra and CTB in particular. Not only are they smaller than WSJ, but the proportion of terms (and tokens) which occur only rarely are much higher.

Chapter 3

Unsupervised partial parsing

In this chapter, I present the core contributions of this research: models for the prediction of non-overlapping grammatical constituents in context – unsupervised partial parsing. The approach recasts the partial parsing problem as a tagging problem, following Ramshaw and Marcus (1995), and shows that standard unsupervised generative probabilistic approaches to sequence modeling – hidden Markov models (HMMs) trained by expectation maximization – are a good approach to this problem. While HMMs produce good results, a better approach, with respect to our evaluation, uses a variation of hidden Markov models called probabilistic right-linear grammars (PRLGs). However, these are less general models than HMMs, and in some environments are more brittle. Finally, a number of different options for the sequence modeling approach are explored, including the capacity of these models to perform a kind of exploratory data analysis or clustering of phrasal types. This chapter contains some content previously reported in Ponvert et al. (2010) and Ponvert et al. (2011).

3.1 Unsupervised partial parsing as a tagging problem

The general hypothesis I bring to the problem of unsupervised partial parsing is that *(local) constituent boundaries can be learned by generalizing on sentence boundaries*. In other words, models to predict local constituent boundaries can be learned by accumulating statistics on the terms which tend to occur at the edges of sentences (and also: terms which specifically tend *not* to occur at the edges of sentences).

The concrete implementation of this idea is to treat unsupervised partial parsing as a tagging problem. This means that the models described in this chapter will actually be designed to predict *tags* (or *labels* or *states*) for each term in context. A sequence of tags for the tokens of a sentence, then, encode a level of constituent structure.

Consider the example in Figure 3.1. The constituents identified in (b) are encoded with per-word tags in (c). The tags used are B, I and O, and the interpretation of these tags is:

B token begins a constituent,

I token is within a constituent,

O token is not within a constituent.

This style of encoding constituent structure as per-token tags was introduced by Ramshaw and Marcus (1995), and variations of this tag-encoding are widely used for supervised chunking and named-entity recogntion (Tjong Kim Sang and Buchholz, 2000; Tjong Kim Sang and De Meulder, 2003; Ratinov and Roth, 2009). Since these tags are specifically intended to encode multi-word constituents, certain rules hold for what possible sequences of tags are allowed. In particular, a B tag may only be followed by an I tag; and an O tag may never by followed by an I tag; so tag sequences beginning with B will be of the form B–I, B–I–I, *etc.* These sequences are decoded as constituents (or chunks).

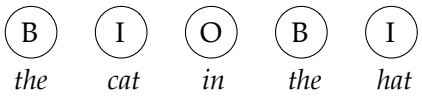
In addition to the B, I and O tags, a STOP tag is used to indicate sentence boundaries, as in Figure 3.1d: a special term # is inserted at each sentence boundary, and this token is consistently given the STOP tag. Additionally, the set of phrasal punctuation alluded to above is manually assigned the STOP tag. Like the O tag, a STOP tag cannot immediately preceded an I tag, and a B tag cannot immediately precede a STOP tag. These constraints on tag sequences enable us to acquire statistics about which terms tend to occur at sentence boundaries. While Figures 3.1c and d give the impression that the tags are known, in an unsupervised learning environment they are not. However, these constraints limit the possible tag sequences available for a sequence of terms, as in Figure 3.1e: only the possible tags are illustrated above each token, and the possible tag sequences are indicated by

the cat in the hat

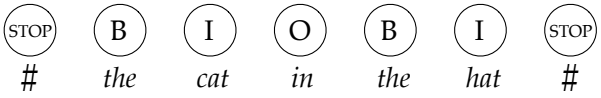
a) Raw text

(*the cat*) *in* (*the hat*)

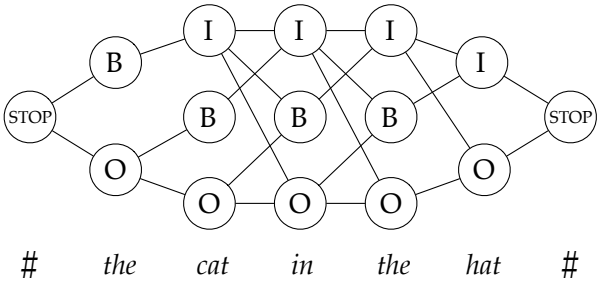
b) Potential constituents for text in (a)



c) BIO tagging for the constituents in (b)



d) BIO tagging with sentence boundaries



e) Possible tag sequences

Figure 3.1: Example of BIO tagging for unsupervised partial parsing.

lines between tags. This figure concisely represents all the possible tag sequences for this example sentence.

Limiting the tag sequences for corpus tokens in this way enables us to learn statistics for which terms tend to occur at sentence boundaries, and which do not. In the example in Figure 3.1e, the term *the* co-occurs with the B tag more frequently than the I tag or the O tag – counting all *possible* tag-token co-occurrences. By learning to give greater weight to the association between the B tag and the term *the*, and by noting that only the I tag can follow the B tag, a statistical model can learn to give greater weight to the association between the I tag and the terms *cat* and *hat*, which are the common nouns in this example. Having learned statistics associating terms and tags, and by learning statistics over tag sequences, we can use these statistics in the form of a *model* to make predictions about the most likely tag sequences for a new seqence of terms.

Statistical models for predicting (unknown) tags for sequences of terms are used in computational linguistics for a variety of applications. One of the most common classes of models – and one that lends itself to unsupervised learning – is the class of *hidden Markov models*, which are described in what follows. A standard form of unsupervised learning of hidden Markov models is *expectation maximization* (EM), which is also described below; intuitively, EM operates by learning statistical assocations in a manner similar to the learning process in the preceding paragraph: by building statistics from possible tag sequences and reinforcing these statistics in an iterative manner.

3.2 Models for unsupervised partial parsing

In this section, the sequence model approach to unsupervised partial parsing is presented. I start by reviewing probabilistic generative finite-state sequence models, *viz.* hidden Markov models, and presenting a variation on HMMs called probabilistic right linear grammars. These models are presented in a general form, without reference to chunking exactly. The connection to chunking comes with the presentation of the label set the models use, and constraints on label (state) transitions which give the labels meaning. The label set is an instance of BIO encoding, variations of which are common in supervised chunking and named entity recognition (NER). This label set encodes a single level of constituent structure as

per-word labels; so, sequence model output in terms of BIO tagging is straightforwardly translated into constituents.

3.2.1 Hidden Markov models

A standard class of probabilistic models for modeling latent information in natural language is hidden Markov models. These have widespread use in natural language processing, and data mining in general, and there are numerous textbook expositions (see, *e.g.*, Rabiner, 1989; Manning and Schütze, 1999; Bishop, 2006; Jurafsky and Martin, 2008; Lin and Dyer, 2010, to cite a few). HMMs are reviewed here for reference and to provide a context for probabilistic right linear grammars, a less familiar variation of HMMs which is used of in this work. Take $\mathbf{x}$ to be a collection of data – say, a multiset of words

$$\mathbf{x} = (\ \#,\ \textit{the},\ \textit{brown},\ \textit{bear},\ \textit{often},\ \textit{sleeps},\ .,\ \ldots\ )$$

drawn from a vocabulary V. In applications like part of speech tagging, the problem is to choose a label for each, *i.e.* to pick a set $\mathbf{y}$ corresponding to $\mathbf{x}$ such that each y_i is the correct label for x_i, as in

$$\begin{aligned}\mathbf{y} &= (\ \text{BOS},\ \text{DET},\ \text{ADJ},\ \text{N},\ \text{ADV},\ \text{V},\ \text{PUNC}\ \ldots\ )\\ \mathbf{x} &= (\ \#,\ \textit{the},\ \textit{brown},\ \textit{bear},\ \textit{often},\ \textit{sleeps},\ .,\ \ldots\ )\end{aligned}$$

where the y_i come from a set T (variously called POS tags, word classes, states, *etc.*; in this example # is taken to represent a special beginning-of-sentence character and BOS a corresponding word class). $\mathbf{x}$ can represent an individual sentence, or alternatively an entire corpus using a special symbol # to indicate sentence boundaries. A generative probabilistic approach to this considers the question: what is the likelihood of a word (or observed variable) w together with a label (hidden state) t, *i.e.* $\Pr(w, t)$. Given a corpus $\mathbf{x}$ and a set of labels $\mathbf{y}$ where each word x_i has an associated label y_i, the full dataset likelihood is given by

$$\Pr(\mathbf{x}, \mathbf{y}) = \prod_i \Pr(x_i, y_i).$$

For practical purposes, such as POS tagging, it is often useful to find the *most likely* set of states for data $\mathbf{x}$, in other words,

$$\underset{\mathbf{y}}{\operatorname{argmax}} \Pr(\mathbf{x}, \mathbf{y})$$

In a mixture model, this is broken into separate probabilities for the conditional probability of the data given the set of labels, and the prior probability of the labels,

$$\Pr(\mathbf{x}, \mathbf{y}) = \Pr(\mathbf{x}|\mathbf{y}) \Pr(\mathbf{y}) \tag{3.1}$$

such that each word and label pair is modeled as a distinct probability

$$\Pr(x_i, y_i) = \Pr(x_i|y_i) \Pr(y_i).$$

A mixture model is a *generative model*, in the sense that it implicitly models a probability distribution over data points x, and so sampling from this distribution yields a set of synthetic data points (Bishop, 2006, p 43). A discrete model, as here, is characterized by a set of parameters θ such that for each $t \in T$, $\Pr(q) = \pi_q$, and for each $q \in T$ and $w \in V$, $\Pr(w|q) = \eta_{q,w}$. So,

$$\begin{aligned} \Pr(\mathbf{x}, \mathbf{y}; \pi, \eta) &= \Pr(\mathbf{x}|\mathbf{y}; \eta) \ \Pr(\mathbf{y}; \pi) \\ &= \prod_i \Pr(x_i|y_i; \eta) \ \Pr(y_i; \pi) \\ &= \prod_i \eta_{y_i,x_i} \ \pi_{y_i} \end{aligned} \tag{3.2}$$

A hidden Markov model generalizes this setup: rather than treating the probabilities of the labels in isolation, each y_i is considered in the context of the what came before it, subject to the Markov assumption. In a first order Markov model, the most recent label is taken as context:

$$\begin{aligned} \Pr(\mathbf{y}) &= \Pr(y_1) \prod_{i>1} \Pr(y_i|y_i, \ldots, y_{i-1}) \\ &\approx \Pr(y_1) \prod_{i>1} \Pr(y_i|y_{i-1}) \end{aligned}$$

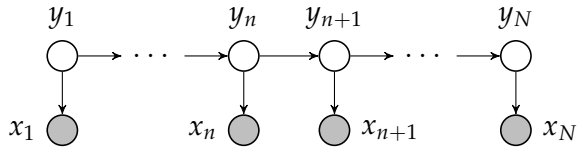


Figure 3.2: HMM graphical model

Modeling the joint probability of the dataset takes the same form as (3.1), only where the likelihood of the states $\Pr(\mathbf{y})$ treats these as a probability of a sequence of states using a Markov model. This is a natural fit for natural language, since of course a sequence of words in a text (or a sentence, *etc.*) have the same linear sequential structure. A first-order HMM, then, has parameters for *state transition* ($B_{q,r}$), *emissions* ($A_{q,w}$) and *initial state probabilities* (π_q), so the data likelihood is calculated by

$$\begin{aligned}\Pr(\mathbf{x}, \mathbf{y}; A, B) &= \Pr(\mathbf{x}|\mathbf{y}; A)\ \Pr(\mathbf{y}; B) \\ &= \left(\prod_i \Pr(x_i|y_i; A)\right) \Pr(y_1; \pi) \left(\prod_{i>1} \Pr(y_i|y_{i-1}; B)\right) \\ &= \left(\prod_i A_{y_i,x_i}\right) \pi_{y_1} \left(\prod_{i>1} B_{y_{i-1},y_i}\right)\end{aligned} \tag{3.3}$$

(3.3) defines a graphical model, which is illustrated in Figure 3.2. This can be simplified a bit: assume that the first observed word (x_1) and the first state (y_1) are fixed and given. For instance, assume that a given string of words starts with the special beginning-of-sentence symbol, #, and corresponding to that there is a special label BOS. Then, initial state probabilities can be dropped from the formula, and dataset likelihood factors simply into emissions and transition probabilities:

$$\Pr(\mathbf{x}, \mathbf{y}; \theta) = \prod_{i>1} A_{y_i,x_i}\ B_{y_{i-1},y_i}$$

A model $\theta = (A, B)$ can be learned via *maximum likelihood estimation* (MLE), that is, by choosing the model $\hat{\theta}$ which maximizes the likelihood of the data. In the case where the labels $\mathbf{y}$ are actually given, as in annotated training material (I use the

notation $\mathbf{y}^\star$ to indicate that labels are provided in this case), then the MLE is

$$\hat{\theta} = \operatorname*{argmax}_{\theta} \Pr(\mathbf{x}, \mathbf{y}^\star; \theta).$$

Intuitively, the maximum likelihood estimates for the HMM parameters may be read from the label counts and word counts,

$$\hat{A}_{q,w} = C(q,w)/C(q)$$
$$\hat{B}_{q,r} = C(q,r)/C(q)$$

where C is the count function:

$$C(q,w) = |\{i : 1 \leq i \leq |\mathbf{x}| \wedge y_i^\star = t \wedge x_i = w\}|$$
$$C(q,r) = |\{i : 1 < i \leq |\mathbf{x}| \wedge y_{i-1}^\star = t \wedge y_i^\star = r\}|$$
$$C(q) = |\{i : 1 \leq i \leq |\mathbf{x}| \wedge y_i^\star = t\}|.$$

When annotated training data is not available, then the MLE of model parameters θ takes the form of a maximization over the marginal probability of the data $\mathbf{x}$:

$$\begin{aligned}\hat{\theta} &= \operatorname*{argmax}_{\theta} \Pr(\mathbf{x}; \theta) \\ &= \operatorname*{argmax}_{\theta} \sum_{\mathbf{y}} \Pr(\mathbf{x}, \mathbf{y}; \theta).\end{aligned}$$

An exact solution for this maximization is intractable for many problems, so a common solution is to use the expectation-maximization (EM) method, an iterative procedure for optimizing the parameters of a joint probability model with latent values, as in this case. The idea is to start with an initial model, θ^{old}, use this model to calculate the likelihood of each possible state t for each word x_i in the dataset $\mathbf{x}$, then use these likelihood estimates to define a new model θ^{new}. θ^{old} is set to the new estimates θ^{new} and this process is iterated, either a set number of times or until it reaches some sort of convergence.

For model re-estimation, in lieu of annotated labels $\mathbf{y}^\star$, *expected counts* are

used:

$$\begin{aligned} \hat{A}_{q,w} &= \frac{\mathbb{E}[C(q,w)]}{\mathbb{E}[C(q)]} \\ \hat{B}_{q,r} &= \frac{\mathbb{E}[C(q,r)]}{\mathbb{E}[C(q)]} \end{aligned} \tag{3.4}$$

These probabilities are estimated by summing over model expectation for each word x_i in the sequence $\mathbf{x}$:

$$\begin{aligned} \mathbb{E}[C(q,w)] &= \sum_{x_i=w} \Pr(y_i = t; \theta) \\ \mathbb{E}[C(q,r] &= \sum_{i>1} \Pr(y_{i-1} = t, y_i = r; \theta) \\ \mathbb{E}[C(q)] &= \sum_{i} \Pr(y_i = t; \theta) \end{aligned}$$

This is the **M-step** of the EM process. The **E-step** corresponds to accurately calculating $\Pr(y_i = t; \theta)$ and $\Pr(y_{i-1} = q, y_i = r; \theta)$ for each i.

For this, two other objects are required: *forward probabilities* α and *backward probabilities* β. The intuition is as follows: $\alpha_{i,q}$ is the sum of the probabilities of all state sequences from 1 to i such that the final state is t, and $\beta_{i,q}$ is the sum of probabilities for all state sequences from i to the end of the dataset $|\mathbf{x}|$ where the first state is t. These are defined recursively:

$$\begin{aligned} \alpha_{1,q} &= \begin{cases} 1 & \text{if } t \text{ is the designated start state} \\ 0 & \text{otherwise} \end{cases} \\ \alpha_{i,q} &= \sum_{r \in T} A_{q,x_i} \, B_{r,q} \, \alpha_{i-1,r} \end{aligned}$$

(If using initial state probabilities, $\alpha_{1,q} = \pi_q \, A_{q,x_1}$.)

$$\begin{aligned} \beta_{|\mathbf{x}|,q} &= 1 \\ \beta_{i,q} &= \sum_{r \in T} A_{r,x_{i+1}} \, B_{q,r} \, \beta_{i+1,r} \end{aligned}$$

Efficiently calculating these values using dynamic programming is done by the

Baum-Welch algorithm. Finally, then, model expectations are calculated as

$$\Pr(y_i = q; \theta) = \alpha_{i,q}\, \beta_{iq}$$
$$\Pr(y_{i-1} = q, y_i = r; \theta) = \alpha_{i-1,q}\, A_{r,x_i}\, B_{q,r}\, \beta_{i+1,r}$$

This presentation leaves out many of the details, and also the proofs that the model estimates above actually do maximize the likelihood of the data; see Rabiner (1989), Jelinek (1998) or the references cited above.[1]

3.2.2 Probabilistic right-linear grammars

HMMs may be thought of as a special case of probabilistic context-free grammars, where the non-terminal symbols are the hidden state space, terminals are the observed states and rules are of the form NONTERM $\rightarrow$ TERM NONTERM. So, the emission and transition emanating from y_n would be characterized as a PCFG rule $y_n \rightarrow x_n\, y_{n+1}$. HMMs factor rule probabilities into emission and transition probabilities:

$$\begin{aligned}\Pr(q \rightarrow w\, r; \theta) &= \Pr(w, r|q; \theta) \\ &= \Pr(w|q; A)\ \Pr(r|q; B) \\ &= A_{q,w}\, B_{q,r}\end{aligned}$$

A probabilistic right linear grammar (PRLG) $\tau = (G, B)$ does not make this independence assumption, and models:

$$\begin{aligned}\Pr(q \rightarrow w, r; G, B) &= \Pr(w, r|q; G, B) \\ &= \Pr(w|q, r; G)\ \Pr(r|q; B) \\ &= G_{q,r,w}\, B_{q,r}\end{aligned}$$

Transition probabilities (B) are still present, but emissions probabilities, now denoted G, are conditioned on the current state and the following state. This way, there is no information loss between the counted observation of rules, *i.e.*

$$C(q\ \rightarrow\ w\, r)$$

[1] This presentation draws from Bishop (2006) and Lin and Dyer (2010) in particular.

and the corresponding probabilities. (Transition probabilities are held separately, as in an HMM, due to constraints stated on them, see below.) See Smith and Johnson (2007) for a discussion, where they refer to PRLGs as Mealy HMMs. For estimating model parameters, EM – and the Baum-Welch algorithm – are exactly the same, only PRLG emissions are additionally conditioned on following state; the HMM arc probability from state q to r at time i, given by $A_{q,x_i}\ B_{q,r}$, is given by a PRLG by $G_{q,r,x_i}\ B_{q,r}$.

Smoothing Deviating slightly from the pure parameter estimation, as in (3.4), *smoothing* is applied to provide non-zero probabilities to unseen terms and to moderate low probabilities of infrequently seen terms, such as *hapax legomena*. Smoothed estimates for HMM (3.5) and PRLG (3.6) emissions are

$$\hat{A}_{q,w} = \frac{C(q,w) + \lambda}{C(q) + \lambda|V|} \tag{3.5}$$

$$\hat{G}_{q,r,w} = \frac{C(q,r,w) + \lambda}{C(q,r) + \lambda|V|}. \tag{3.6}$$

In unsupervised estimation, $C(q,w)$ and the others stand for expected counts, or strictly $\mathbb{E}[C(q,w)]$. $C(q,r,w)$ is the (expected) count of $x_i = w$ where $y_i = q$ and $y_{i+1} = r$. $|V|$ is the number of terms w, and λ is a smoothing parameter. λ is set to .1 for all experiments; more sophisticated smoothing could avoid dependence on a parameter for smoothing. This value was chosen since it produced good results across the datasets, based on early preliminary evaluation using the development datasets.

3.2.3 Chunking with sequence models

Labels for chunking The chunkers evaluated in this work are constrained sequence models (HMMs and PRLGs) over label sets encoding local constituent structure (Ramshaw and Marcus, 1995). So, rather than labels corresponding to parts of speech or word classes in the conventional sense, labels assigned to words encode a level of constituent structure. A simple set of labels for this is BIO, which is familiar from supervised chunking and named-entity recognition: the tag B denotes the beginning of a chunk, I denotes membership in a chunk and O denotes exclusion

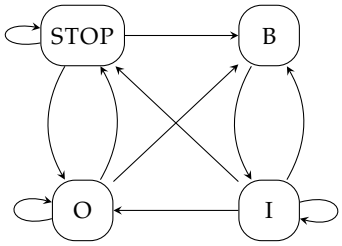


Figure 3.3: Possible tag transitions as a state diagram.

	STOP	B	I	O
STOP	.33	.33		.33
B			1	
I	.25	.25	.25	.25
O	.33	.33		.33

Figure 3.4: Uniform initialization of transition probabilities subject to the constraints in Figure 3.3: rows correspond to antecedent state, columns to following state.

from any chunk. In addition the tag STOP is associated with sentence boundaries and phrasal punctuation.

Constraints characterizing chunking Transition probabilities are not smoothed in this work, so

$$\hat{B}_{q,r} = C(q,r)/C(q) \tag{3.7}$$

for both HMMs and PRLGs. This is for two reasons. First, with four tags, there is no data-sparsity concern with respect to transitions. Second, the nature of the task imposes certain constraints on transition probabilities: because multiword chunks, specifically are targeted in this application, B cannot follow a B – in other words $B_{\text{B,B}} = 0$.

These constraints boil down to the observation that the B and I states will only be seen in BII* sequences. This may be expressed via the state transition diagram in Figure 3.3.

In EM learning, if in some model θ^{old} a parameter is 0 – some $A_{q,w}$ or $B_{q,r}$, *etc.* – then in the next iteration θ^{new}, this parameter will remain 0 (provided smoothing is not applied to that parameter estimation). Thus, a simple way to enforce the

constraints specified implicitly in Figure 3.3 is to set *initial* model parameters for these transitions to 0.

That said, the initial model parameters are uniform distributions subject to the constraints of Figure 3.3. These are given in Figure 3.4. I also experimented with random initial models (subject to the constraints in Figure 3.3). Uniform initialization usually works better; also, uniform initialization does not require multiple runs of each experiment, as random initialization does.

3.2.4 Motivating the HMM and PRLG.

This approach – encoding a chunking problem as a tagging problem and learning to tag with HMMs – goes back to Ramshaw and Marcus (1995). For unsupervised learning, the expectation is that the model will learn to generalize on phrasal boundaries. That is, the models will learn to associate terms like *the* and *a*, which often occur at the beginnings of sentences and rarely at the end, with the tag B, which cannot occur at the end of a sentence. Likewise common nouns like *company* or *asset*, which frequently occur at the ends of sentences, but rarely at the beginning, will come to be associated with the I tag, which cannot occur at the beginning.

The basic motivation for the PRLG is the assumption that information is lost due to the independence assumption characteristic of the HMM. With so few states, it is feasible to experiment with the more fine-grained PRLG model.

3.3 Evaluation

The primary evaluation for the methods proposed here is direct comparison to gold standard constituents from treebanks. The gold standards used are constituent chunks and base NPs, described earlier.

Data preparation and CCL benchmark Like Seginer (2007a), text is lower-cased but otherwise not altered. Sentence segmentation and tokenization from the treebank is used. These experiments make use of phrasal punctuation, as described above, as heuristic indicators of phrasal boundaries. The benchmark used for comparison is low-level (non-nested) constituents extracted from CCL parser output – this benchmark is referred to as 'CCL*' in the results tables and discussion below.

CCL was run with the same experimental setup in terms of data preparation and use of punctuation. This is also the same data preparation setup used in previous work by Seginer (2007a,b). For the constituent chunking and base NP identification tasks, precision, recall and *F*-score are reported for full constituent identification – brackets which do not match the gold standard exactly are false positives.

One of the motivations of UPP in this work is to improve upon the local (non-nested) constituent predictions of general unsupervised parsers, such as Klein and Manning's CCM or Seginer's CCL. Lowest-branching constituents are extracting from parser output and evaluated in the constituent chunking, base NP identification and treebank precision evaluations outlined in Chapter 2.7.2.

Because CCL was not really intended to be evaluated this way, an alternative evaluation using parser output is given below: recall of parser output on the chunking and base NP gold standards, respectively. That is, using the subset of gold standard constituents used for constituent chunking and base NP, what proportion of each subset were identified by the parser.

Gold-standard POS based benchmark For the constituent chunking and base NP identification tasks, model results are also compared to the simple supervised benchmark based on matching sequences of gold standard POS, described in Chapter 2.7.2. The idea is to collect a set of POS tag sequences – using gold-standard POS, which of course neither CCL or the sequence models have access too – then identify potential sequences of POS in the evaluation material which correspond to a known POS sequence collected from training. To illustrate, the 5 most frequent POS sequences corresponding to NPs in each of the datasets (from the training subset) is given in Table 3.1, together with the proportion of total NPs in the (training) corpus represented by the sequence. Take the POS tag sequences for WSJ, for example. Given these, for a sentence represented by

```
DT NNP POS NNP NNP RB VBD DT NNS
```

the heuristic would pick out two sequences – `NNP NNP` and `DT NNS` – thus chunking the string as

```
DT NNP POS (NNP NNP) RB VBD (DT NNS)
```

WSJ		Negra		CTB	
DT NN	17.8	ART NN	28.1	NN NN	28.7
NNP NNP	7.4	ART ADJA NN	9.1	NR NN	8.4
DT JJ NN	6.0	ADJA NN	6.1	JJ NN	7.9
JJ NNS	4.8	PPOSAT NN	2.9	NN NN NN	5.1
DT NNS	3.1	CARD NN	1.7	DT NN	3.2
Total	39.1	Total	48.0	Total	53.5

Table 3.1: Top 5 POS sequences in the training datasets corresponding to NPs. For the POS tagset for WSJ, see Marcus et al. (1993); Negra uses the Stuttgart/Tübingen Tagset, see Brants (2000); for CTB see Xia (2000).

	WSJ	Negra	CTB
HMM	41	56	53
PRLG	78	181	78

Table 3.2: Number of iterations required before models converged in EM.

In the results that follow, GPOS-5 refers to this benchmark matching the top-5 most frequent POS sequences for constituent chunks and base NPs respectively. This benchmark do not use phrasal punctuation.

Model setup The HMM and PRLG models were initialized uniformly as illustrated in Figure 3.4. Sequence model parameters are interactively refined via EM using the training data. I report accuracy only after convergence, that is after the change in full dataset perplexity (log inverse probability) is less than .01% (.0001) between iterations. The number of iterations required before each model converges is given in Table 3.2.

3.3.1 Primary results on held-out test datasets

I begin by presenting the core results of the models' evaluation on held-out test data; detailed results and analysis using development datasets follow.

Model performance results on held-out test datasets are reported in Table 3.3 and Table 3.4. The sequence models outperform the CCL* benchmark at both tasks and on all three datasets. In most cases, the PRLG sequence model performs better than the HMM; the exception is CTB, where the PRLG model is behind

	WSJ			Negra			CTB		
	Prec.	Rec.	*F*	Prec.	Rec.	*F*	Prec.	Rec.	*F*
GPOS-5	*62.5*	*33.6*	*43.7*	*54.9*	*35.1*	*42.9*	*53.8*	*36.5*	*43.5*
CCL*	57.5	53.5	55.4	28.4	29.6	29.0	23.5	23.9	23.7
HMM	53.8	62.2	57.7	35.0	37.7	36.3	**37.4**	**41.3**	**39.3**
PRLG	**76.2**	**63.9**	**69.5**	**39.6**	**47.8**	**43.3**	23.0	18.3	20.3

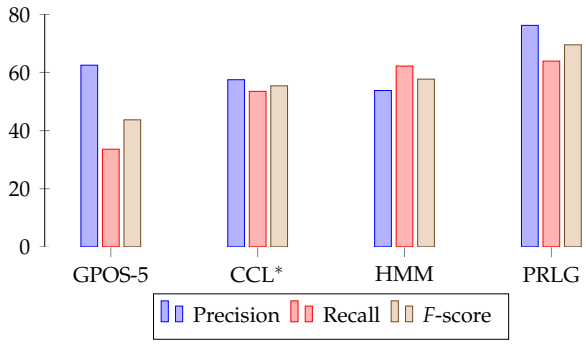
Table 3.3: Unsupervised constituent chunking results. GPOS-5 indicates the benchmark of identifying constituent chunks by the 5 most common gold standard POS sequences; CCL* indicates baseline of chunks extracted from CCL parser output.

	WSJ			Negra			CTB		
	Prec.	Rec.	*F*	Prec.	Rec.	*F*	Prec.	Rec.	*F*
GPOS-5	*62.6*	*40.1*	*48.9*	*34.8*	*47.6*	*40.2*	*53.4*	*41.1*	*46.4*
CCL*	46.2	51.1	48.5	15.6	29.2	20.3	10.4	17.3	13.0
HMM	47.7	65.6	55.2	23.8	46.2	31.4	17.0	**30.8**	21.9
PRLG	**76.8**	**76.7**	**76.7**	**24.6**	**53.4**	**33.6**	**21.9**	28.5	**24.8**

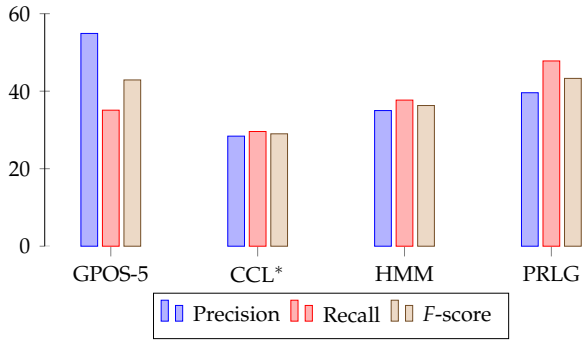
Table 3.4: Unsupervised NP identification results. GPOS-5 is targeting base NPs rather than constituent chunks. For CCL*, HMM and PRLG, the same model output from Table 3.3 evaluated.

	WSJ	Negra	CTB
CCL	57.5	34.1	37.2
CCL^{NR}	50.4	26.5	31.2
CCL*	59.2	23.4	31.7
HMM	55.4	35.1	43.4
PRLG	83.2	40.4	43.8

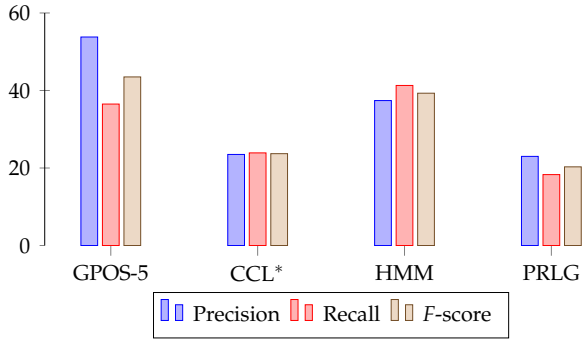
Table 3.5: Precision of CCL, CCL output not counting the root constituent (CCL^{NR}), chunks extracted from CCL (CCL*), and models for UPP on precision of predicting treebank constituents.



a) WSJ

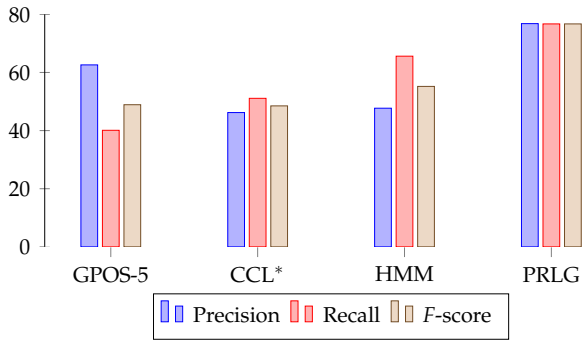


b) Negra

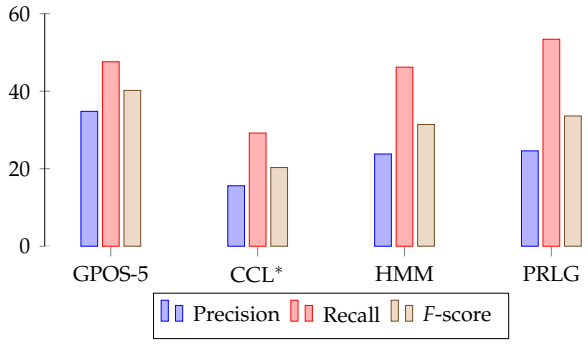


c) CTB

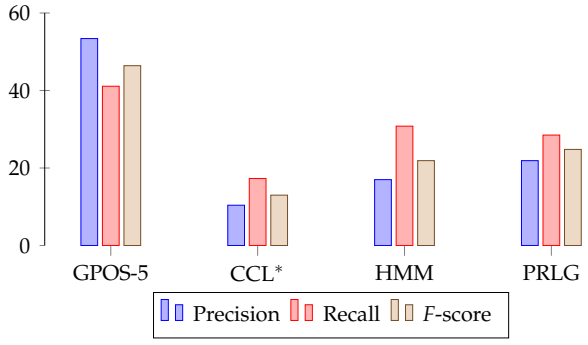
Figure 3.5: Bar charts for baselines and models for constituent chunking.



a) WSJ



b) Negra



c) CTB

Figure 3.6: Bar charts for baselines and models for base NP identification.

the HMM in evaluation, as well as behind CCL.

For WSJ, both sequence models often also beat the GPOS-5 benchmark on both tasks; also for Negra the PRLG model beats this baseline for the constituent chunking task. For WSJ and CTB, the GPOS benchmarks do better at base-NP identification than constituent chunking. This makes sense, since base NPs are defined by phrasal category whereas constituent chunks are defined by structural tree configuration, so it seems natural that there would be a more tangible relationship between base NPs and the POS they typically occur with.

For Negra, the lower performance of GPOS strategies on NP identification may be related to the fact that NPs within prepositional phrases are not annotated. The sequence models also show relatively low precision on NP identification against Negra. Here, the sequence model chunkers often do find NPs embedded in PPs, which are not annotated as such. For instance, in the PP "hinter den Kulissen" (behind the scenes), both the PRLG and HMM chunkers identify the internal NP, though this is not identified in Negra and thus considered a false positive. The fact that the HMM and PRLG have higher recall on NP identification on Negra than precision is further evidence towards this.

The results of these evaluations are displayed graphically for constituent chunks in Figure 3.5 and for base NP identification in Figure 3.6.

Precision on all treebank constituents Evaluation results for the sequence models by precision on all treebank constituents is given in Table 3.5. That is, what proportion of all treebank constituents are identified by the different models. These are compared to two benchmarks in these experiments: the precision of CCL, and the precision of the chunks extracted from CCL parser output (CCL*). Normally, local parser predictions have higher precision than non-local (longer) constituent predictions. However, in this setup, evaluation of parser output generally counts the root node – the bracket surrounding the whole sentence – as a true positive; evaluation of (UPP) chunker output, including CCL* (the chunks extracted from CCL parser output) basically do not get a true-positive for free in this manner, thus these numbers are inflated. Table 3.5 also provides the precision of all CCL-output brackets except the root bracket – noted CCL$^{\text{NR}}$. So, expecting CCL to have generally higher precision at low-level predictions than higher constituent predictions, except the root constituents, CCL* precision will likely be higher than the CCL$^{\text{NR}}$

number; this is the case for WSJ and CTB, but curiously not for Negra. This is possibly related to the issue with Negra raised earlier, that NPs within PPs are not annotated as constituents, thus CCL predictions of these constituents would be counted as false positives.

Compared to these different views of CCL on the treebank-precision evaluation, the sequence models almost always outperform all of them. The one exception to this is that the CCL* brackets – the local predictions of CCL – on WSJ have higher precision than the HMM unsupervised chunking model. In every case, the best-performing model is the PRLG; the PRLG performs dramatically better on WSJ, beating the CCL* benchmark by 24 percent.

Recall of chunks and base NPs In Negra base-NP identification evaluation, numerous actual noun phrases within prepositional phrases are not annotated as such in the treebank. With that in mind, the operative metric for Negra base NP identification is recall: of the NPs that are annotated as such, what proportion were identified by model. Looking specifically at recall, both sequence models outperform the CCL* and the GPOS-5 baselines: see Figure 3.6b.

However, looking just at recall puts the CCL* baseline at a disadvantage, since really the CCL parse was not intended to be evaluated in this way. Thus, Table 3.6 compares the constituent chunk and base NP recall metrics of the sequence models to the recall of *all* constituents predicted by CCL. Even compared to this, the sequence models' recall is almost always higher.

3.3.2 Development results and benchmarks

In this section I report model results on the development datasets used for exploration. These are the datasets and numbers which were used primarily to evaluate the models during the development of this research; also these datasets are also used for deeper data analysis and comparisons than the held-out test datasets, which are reserved for final empirical comparison with the benchmark. In addition to the sequence models and the GPOS-5 and CCL benchmarks, the results on development datasets reported in Table 3.7 and Table 3.8 include two more benchmarks for comparison.

	WSJ	Negra	CTB
CCL	57.8	36.0	25.5
HMM	62.2	37.7	**41.3**
PRLG	**63.9**	**47.8**	18.3

a) Recall of constituent chunking

	WSJ	Negra	CTB
CCL	57.8	38.8	23.2
HMM	65.6	46.2	**30.8**
PRLG	**76.7**	**53.4**	28.5

b) Recall of base NPs

Table 3.6: Recall of CCL on the constituent chunk and base NP identification tasks.

GPOS-$\star$ benchmark A stronger version of the GPOS-5 benchmark takes the best result from the top-N collections. This benchmark is referred to as GPOS-$\star$. That is, I experimented with the top-5 most frequent POS sequences, the top-10, -25, -50, -100, -250 and top-500 most frequent POS sequences; GPOS-$\star$ represents the best result from any of these sets. These benchmarks model chunking strategies which makes use of POS or word-classes, but have access to gold-standard POS, have access to knowns constituent chunks and/or base NPs in the training data, and – in the case of GPOS-$\star$ – roughly knows the optimal set of POS sequences for this strategy. The latter turns out to be a fairly strong benchmark.

Supervised benchmark As another benchmark, the sequence models are run in a *supervised* manner – that is, they have access to the gold standard chunks and/or base NPs, these were converted to a tagged corpus using the BIO conventions outlined above, then model parameters were estimated using maximum likelihood estimation. That is, model parameters are estimated just as they are in (3.5), (3.6) and (3.7), only the count function $C(\cdot)$ refers to observation counts of events from the gold-standard training, rather than soft counts from the Baum-Welch trellis. The idea here is to consider how close parameters learned via unsupervised (EM) estimation come to the presumably near-ideal parameters learned from annotated training material. This should not be taken to represent a state-of-the art for supervised chunking or partial parsing. Neither this benchmark nor the GPOS benchmarks use phrasal punctuation.

Results Results of the CCL benchmark, the GPOS benchmarks, the supervised benchmarks, and the sequence models are reported in Tables 3.7, 3.8 and 3.9.

The different benchmark figures help distinguish the different tasks. For English both the GPOS-based benchmarks and the supervised models perform better than the base NPs task than at the constituent chunking task. However, for Negra, all the benchmarks perform worse at the base NP identification task; and for CTB results are mixed: the *GPOS-⋆* benchmark performs better at base NP identification, whereas the other benchmarks perform better at constituent chunking.

These also provide another point of comparison with the proposed sequence models for unsupervised chunking. For WSJ chunking, the PRLG model in particular performs comparatively well, outperforming the supervised HMM benchmark on constituent chunking and base NP identification. The unsupervised PRLG chunker also outperforms both GPOS benchmarks. The unsupervised models are also reasonably good at Negra chunking as well, with the PRLG model results relatively close (within 5 points *F*-score) to the supervised HMM benchmark. Results are not as positive for CTB chunking, however. I elaborate on these results in following sections (p. 72 in the next section and §3.3.8 on p. 92). Overall, though, the problem may not be so much that the HMM and PRLG are bad models for identifying constituents in Chinese, but rather that constituent chunking and base NP identification – using CTB annotations – are not evaluations which overlap with model predictions. *Precision on all treebank constituents* results in Table 3.9, on the other hand, show that the sequence models are learning to predict constituents with relatively high precision – compared to the CCL baseline – but these constituents are evidently neither constituent chunks nor base NPs.

3.3.3 Learning curves

The final evaluation numbers given in Table 3.3, Table 3.4 and those following only give one picture of model performance: using the set of training data available, with EM run to convergence. This ignores a variety of details about this approach: how much data is sufficient to learn reasonably models for unsupervised chunking? how quickly does the learning process converge to a stable model?

Learning curves are a visualization of classifier model performance at different stages in the classifier's learning process. Usually, learning curves graph

	WSJ			Negra			CTB		
	Prec.	Rec.	*F*	Prec.	Rec.	*F*	Prec.	Rec.	*F*
GPOS-5	*62.4*	*31.2*	*41.6*	*54.5*	*37.1*	*44.1*	*53.3*	*37.3*	*43.9*
GPOS-⋆	*58.1*	*63.5*	*60.6*	*65.4*	*87.6*	*74.9*	*43.6*	*62.6*	*51.4*
Sup-HMM	*69.9*	*67.5*	*68.7*	*46.1*	*48.5*	*47.3*	*60.0*	*59.2*	*59.6*
Sup-PRLG	*77.2*	*75.0*	*76.1*	*56.7*	*60.3*	*58.4*	*63.5*	*63.6*	*63.6*
CCL*	56.2	52.7	54.4	30.0	31.0	30.5	23.1	23.7	23.4
HMM	52.8	61.5	56.8	33.9	39.5	36.5	**36.6**	**41.4**	**38.9**
PRLG	**76.1**	**63.7**	**69.3**	**39.6**	**51.1**	**44.6**	22.7	18.5	20.4

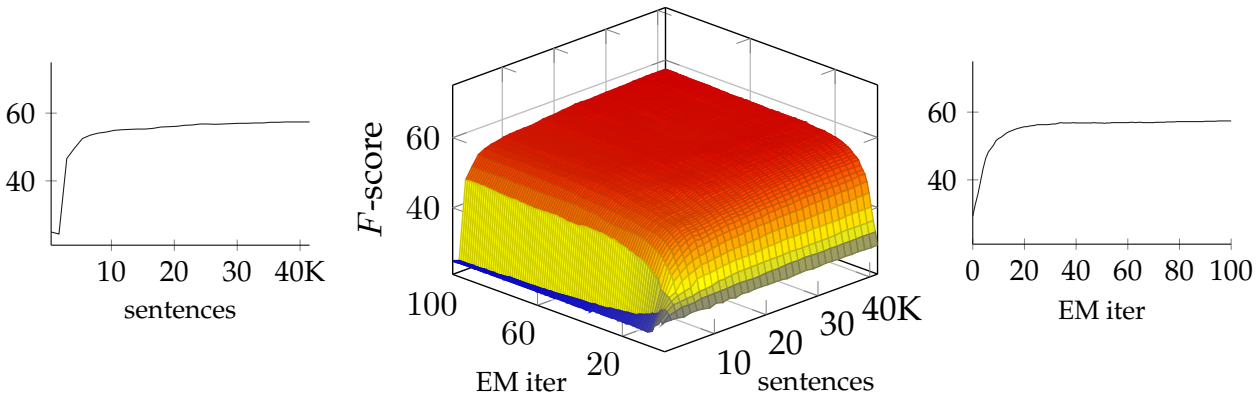
Table 3.7: Unsupervised constituent chunking results on the development datasets. GPOS-5 indicates the benchmark of identifying constituent chunks by the 5 most common gold standard POS sequences; GPOS-⋆ indicates the best-performing POS-matching benchmark for identifying NPs; Sup-HMM and Sup-PRLG are supervised benchmarks: the HMM and PRLG are trained from gold-standard constituent chunks, extracted from the training datasets. CCL* indicates baseline of chunks extracted from CCL parser output.

	WSJ			Negra			CTB		
	Prec.	Rec.	*F*	Prec.	Rec.	*F*	Prec.	Rec.	*F*
GPOS-5	*63.2*	*38.0*	*47.4*	*38.5*	*49.4*	*43.3*	*56.2*	*45.0*	*49.9*
GPOS-⋆	*73.3*	*76.5*	*74.9*	*41.7*	*65.4*	*50.9*	*50.2*	*55.1*	*52.5*
Sup-HMM	*75.9*	*75.4*	*75.7*	*40.5*	*34.3*	*37.2*	*45.4*	*40.7*	*42.9*
Sup-PRLG	*82.4*	*82.3*	*82.4*	*46.4*	*48.3*	*47.3*	*51.9*	*54.6*	*53.2*
CCL*	45.3	50.5	47.8	17.2	30.9	22.1	10.2	16.7	12.7
HMM	47.2	65.3	54.8	23.3	46.8	31.1	17.4	**31.4**	22.4
PRLG	**76.9**	**76.5**	**76.7**	**24.5**	**54.6**	**33.8**	**21.8**	28.3	**24.6**

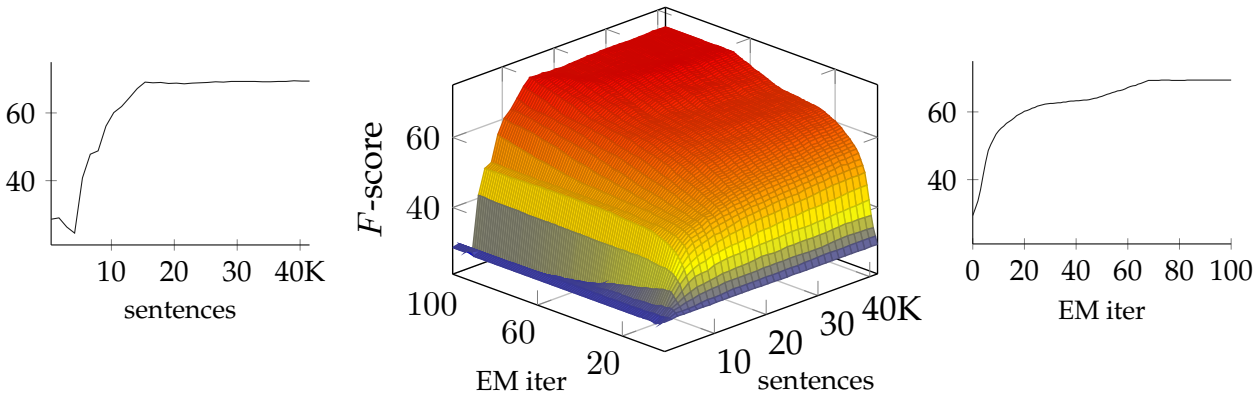
Table 3.8: Unsupervised NP identification results on the development datasets. GPOS-5 and GPOS-⋆ are similar to those in Table 3.7, only targeting base NPs rather than constituent chunks; Sup-HMM and Sup-PRLG are supervised benchmarks, trained with gold standard base NPs extracted from training dataset. For CCL*, HMM and PRLG, the same model output from Table 3.7 is evaluated.

	WSJ	Negra	CTB
CCL	52.4	34.4	37.8
HMM	54.5	33.9	42.8
PRLG	83.1	40.2	44.8

Table 3.9: Treebank precision evaluation numbers from the development datasets.

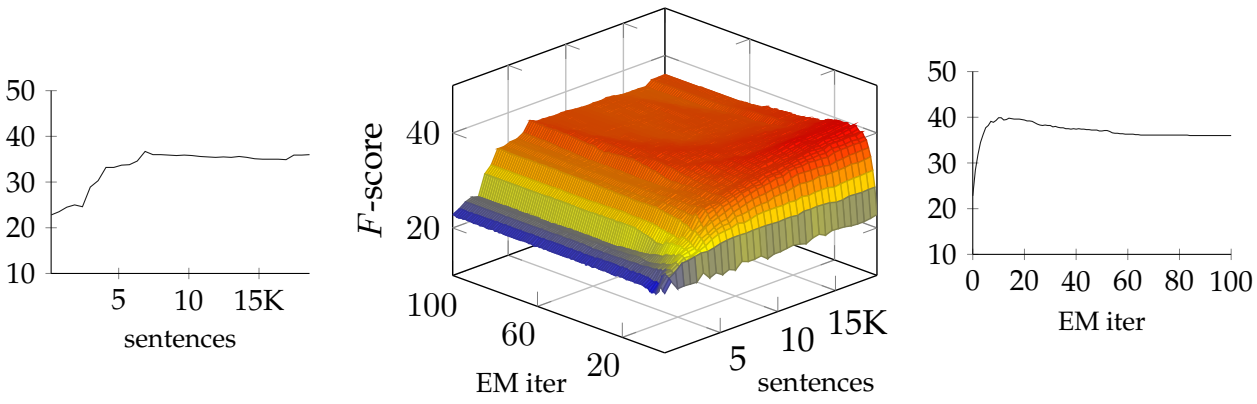


a) HMM chunker (*F*-score)

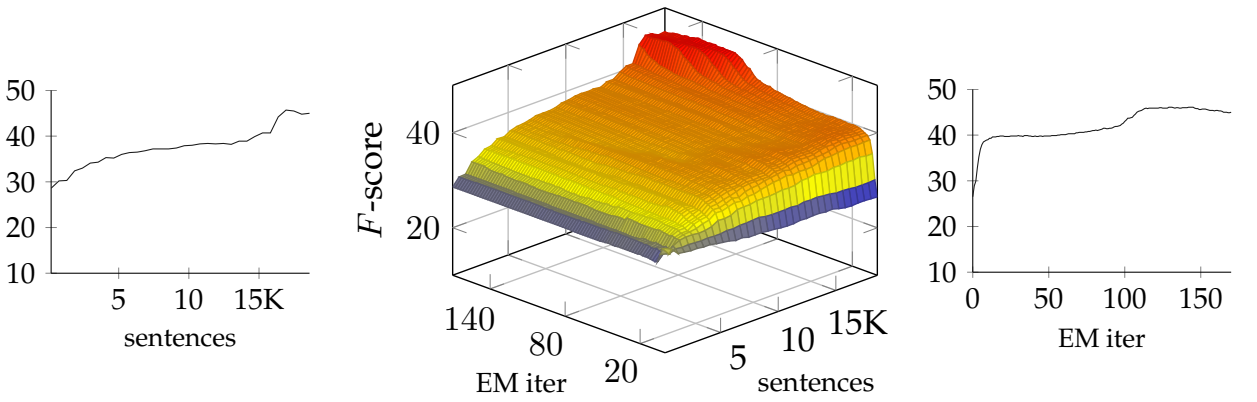


b) PRLG chunker (*F*-score)

Figure 3.7: Learning curves for constituent chunking: WSJ

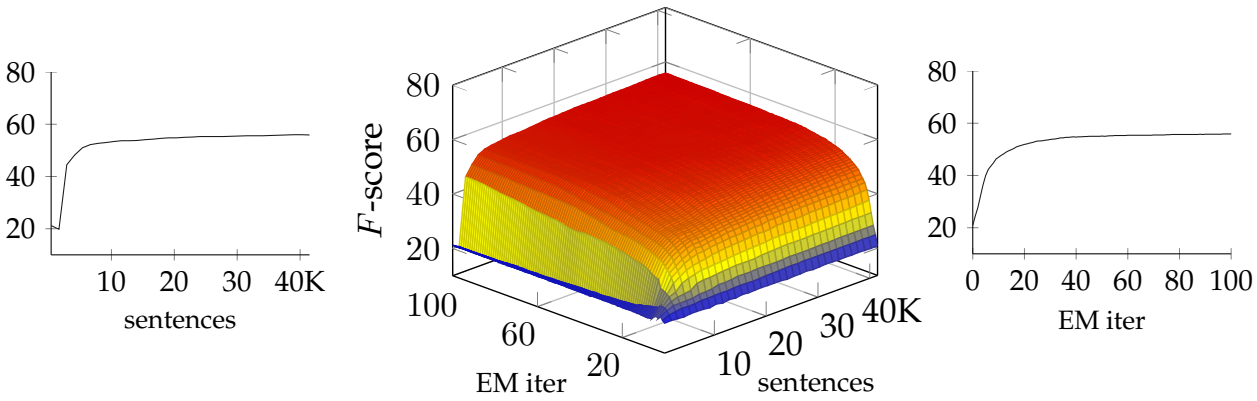


a) HMM chunker (F-score)

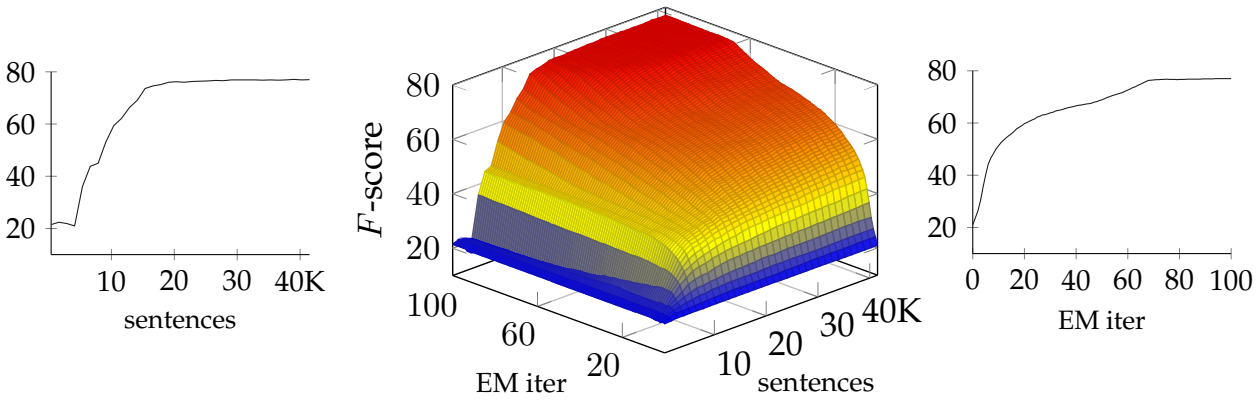


b) PRLG chunker (F-score)

Figure 3.8: Learning curves for constituent chunking: Negra

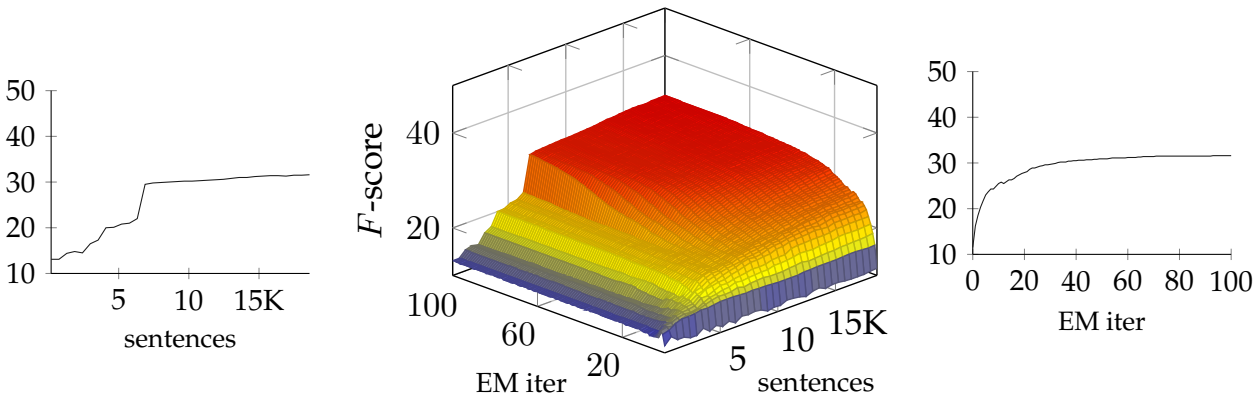


a) HMM chunker (F-score)

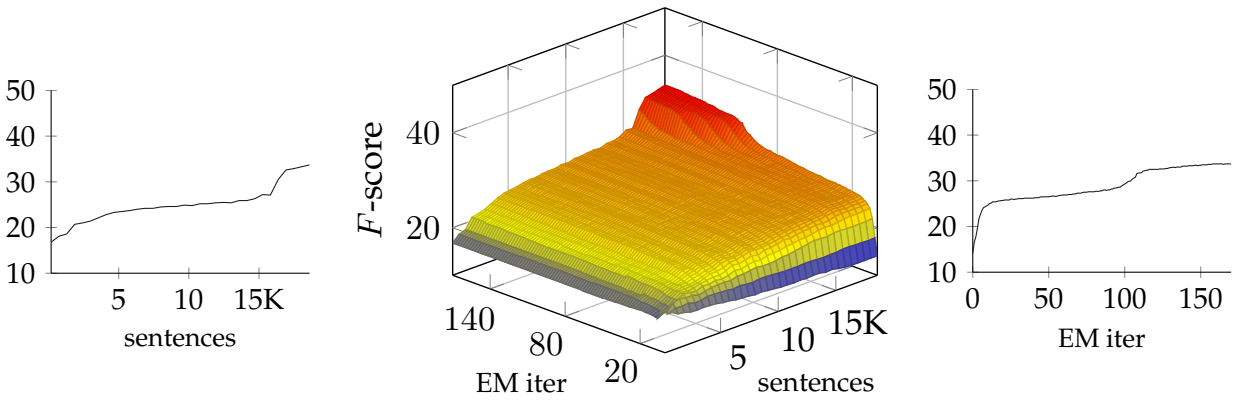


b) PRLG chunker (F-score)

Figure 3.9: Learning curves for base NP identification: WSJ

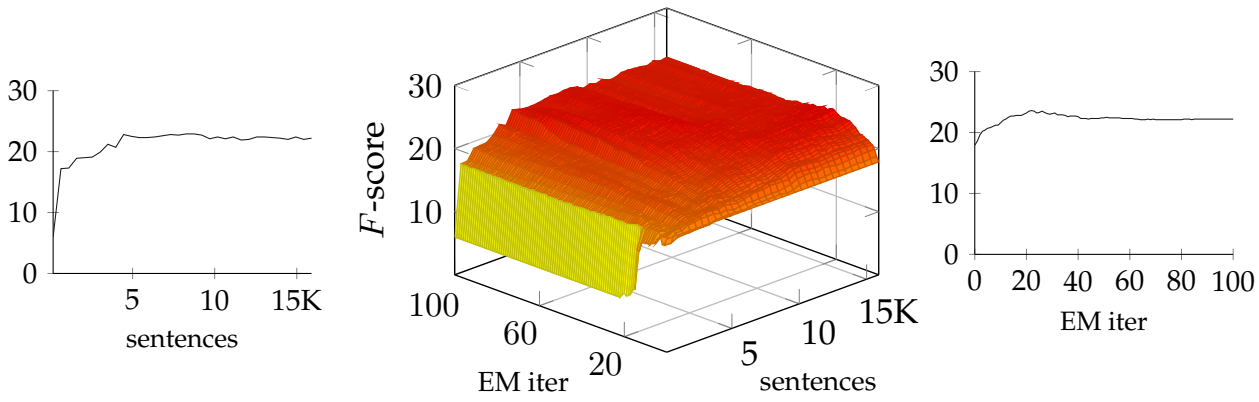


a) HMM chunker (F-score)

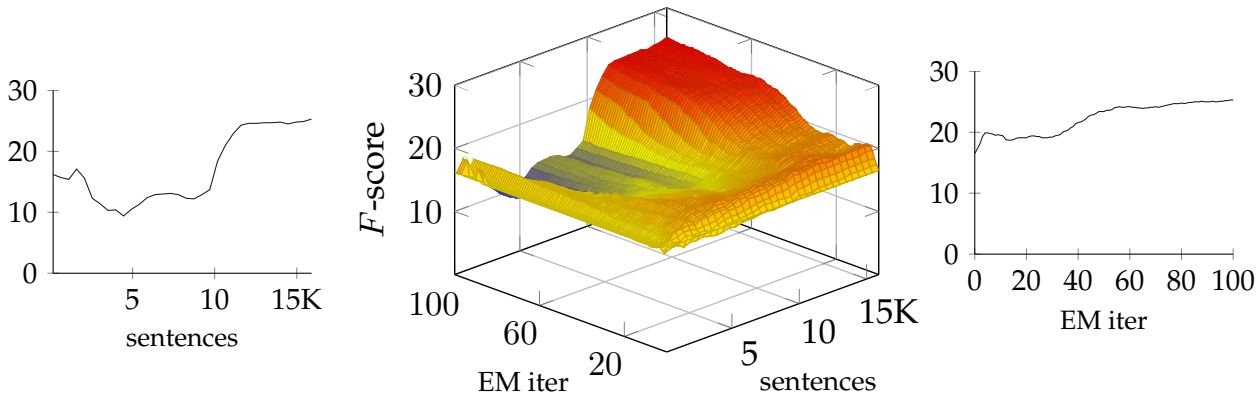


b) PRLG chunker (F-score)

Figure 3.10: Learning curves for base NP identification: Negra



a) HMM chunker (*F*-score)



b) PRLG chunker (*F*-score)

Figure 3.11: Learning curves for base NP identification: CTB

classifier performance relative to the *amount of training material* used to train the classifier. In a two-dimensional graph, this means typically the amount of training material – here, the number of sentences available to the learning process – is the x-axis, and model performance – here, F-score on constituent chunking and base NP identification – is the y-axis. However, there is at least one other dimension to consider: model performance at different stages of the EM learning process. That is, a two dimensional learning curve would have the number of iterations of EM as the x-axis, as in Figure 3.20.

The learning curves in figures 3.7 to 3.11 display both. Learning curve graphs are provided for constituent chunking, base NP identification using both the volume of training data (in sentences) and the number of EM iterations as parameters. These curves are graphed jointly (the three-dimensional graphs), but the overall F-score learning curve over training and the curve over EM iterations are also provided – the edges of the three dimensional curves, essentially.

These are based on the development set results of §3.3.1. The test subsets of the datasets are held constant for each evaluation; the training data is broken into subsets by individual sentences. The experiments consider successively larger training datasets, each of which is the first N sentences of the full training set for some N. One contrast with the experiments from §3.3.1 is that each evaluation is run for 100 iterations of EM, rather than run to convergence; this usually means that EM was run for longer than in the experiments above – most experiments take between 50 to 100 iterations to converge. Otherwise, the parameters – the BIO tagset, the smoothing parameters and model initialization – are the same as in §3.3.1.

Constituent chunking curves are omitted for CTB, in part because these results are discussed in §3.3.8. Also, as discussed there, constituent chunking results degrade over the course of EM, largely because the constituents predicted are longer than those targeted by this evaluation. However, base NP identification results for CTB are more easily visualized and interpreted, and these are given in Figure 3.11. In these curves, the HMM clearly converges more quickly than the PRLG, though the PRLG ultimately has better performance on this task. Though, to echo an earlier point, this evaluation does not reflect the grammatical structure of CTB these models are learning as well as the treebank precision metric.

Assessment For WSJ constituent chunking and NP identification, the PRLG stands out as a better model in these visualizations. However, its learning curves are not the smooth curves for the HMM chunkers. More so than the HMM, the PRLG models learn more from this dataset by allowing EM to run longer.

For Negra constituent chunking and base NP identification, the HMM based chunker actually seems to be the best model overall, even though in the final results – the full training set, EM run to convergence, *c.f.* tables 3.3 and 3.4 – the PRLG chunker beats the HMM in these evaluations. Indeed, for those results, the HMM converges at 56 iterations, whereas the PRLG does not converge until 181 iterations (this is the only evaluation in the experiments to do so, *c.f.* Table 3.2. Some of the best performance on the Negra constituent chunking task comes relatively early in the EM learning process, specifically for the HMM chunker learner. This pattern is not reflected in the base NP identification analysis, however.

For CTB, the PRLG is ultimately the superior model for base NP identification, but only after most of the training data is consumed – at about 10K sentences. Note, this is still a very small amount of textual data, compared to most textual datasets. In contrast, the HMM converges quite quickly on this data, both with relatively little training data, and in terms of iterations of EM.

Learning curves with supervised benchmarks It is useful to compare learning curves plotting the performance of unsupervised methods with supervised variations. This allows us to visually inspect how the models change with different amounts of training. This also allows us to form an intuition about how much human-made annotation is necessary to achieve comparable performance to the unsupervised method.

Figure 3.12 plots learning curves for both the supervised and unsupervised PRLG taggers for base NP identification (F-score). For both unsupervised and supervised NP identification, the PRLG models were either the best performing or close to it. The unsupervised models are trained, as before, using EM run to convergence, whereas the supervised models do not refine their model estimates using EM or any other iterative procedure: they are simply trained via (smoothed) maximum likelihood estimation on the gold standard annotations in the training dataset.

It turns out that the supervised models achieve the performance which the

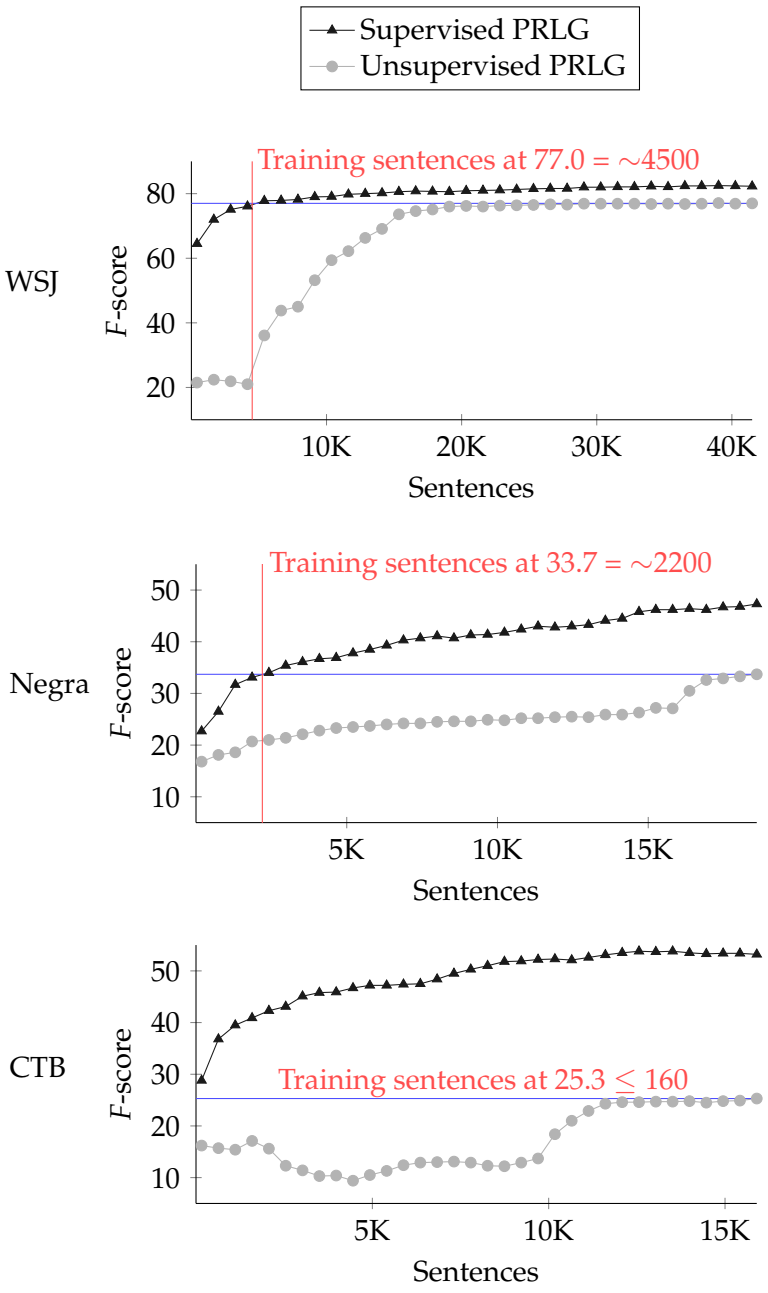


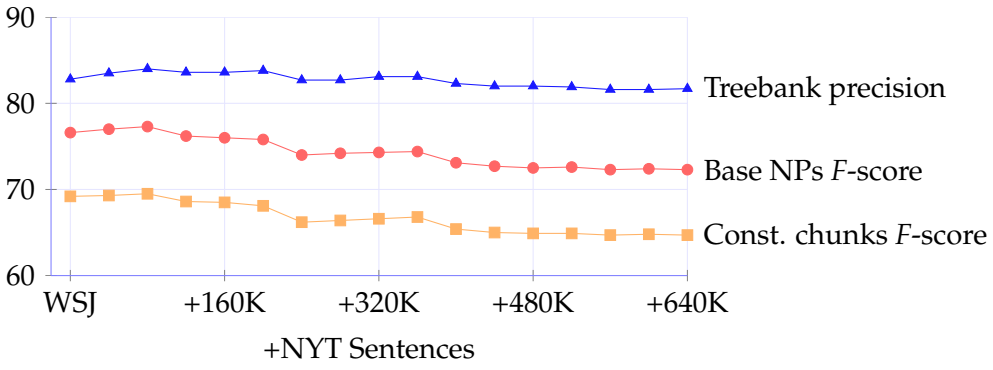
Figure 3.12: Supervised and unsupervised learning curves for NP identification.

unsupervised models achieve with the full training dataset fairly early in the learning curve: for WSJ this is about 4400 sentences, for Negra this is about 2200 sentences. Though this is a fraction of the training data available in these experiments, it is worth noting that annotating 2000 or 4000 sentences can require a significant human effort, even annotating for base noun phrases. For CTB, the supervised model trained with only $\sim$160 sentences has better *F*-score in evaluation than any of the unsupervised-trained models, though see §3.3.8 below: base NP identification evaluation does not provide the best perspective on sequence model chunking for Chinese. Of course, these experiments only reflect the sequence models described in this chapter trained in a supervised manner, and do not represent a state-of-the-art approach to chunking (see *e.g.* Sha and Pereira, 2003).

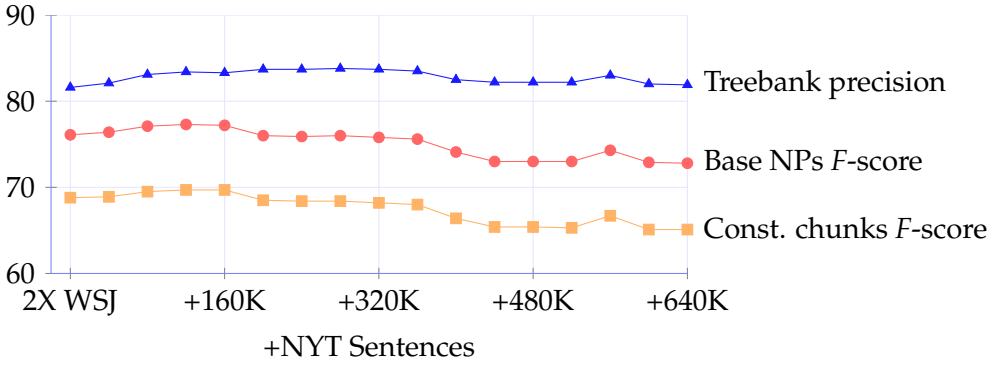
Scaling beyond the training set Since these are unsupervised methods, and labeled annotations are not required for learning models, there is no reason why results should be confined to models trained using data from the treebanks used in evaluation. This facilitates comparison with other published results, but does not reflect a realistic application of unsupervised methods. For this reason, I experimented with adding more data from a similar domain: news text from the New York Times (NYT) section of the English Gigaword corpus.[2] The NYT text is preprocessed using the OpenNLP sentence detector and tokenizer[3] which produces output similar to Penn Treebank conventions (*e.g.* *n't* and *'s* are treated as terms). The best performing model for WSJ chunking – the PRLG – is trained on the combined WSJ + NYT datasets and evaluated using the development dataset. Also, in a variation from the experiments reported above, a smaller threshold for convergence for EM is used: models stop when change-in-data perplexity is $< 10^{-5}$, before it was 10^{-4}; this led to slightly better results with the auxiliary training material, and more consistent results from step to step in the learning curves.

Figure 3.13a illustrates the effect of adding successively larger amounts of additional textual data to the training set, in addition to the original WSJ training set. Reported in that figure are constituent chunking *F*-score, base NP identification *F*-score and precision at identifying treebank constituents. Each step in the plots corresponds to adding 40K sentences of NYT data, which is approximately the

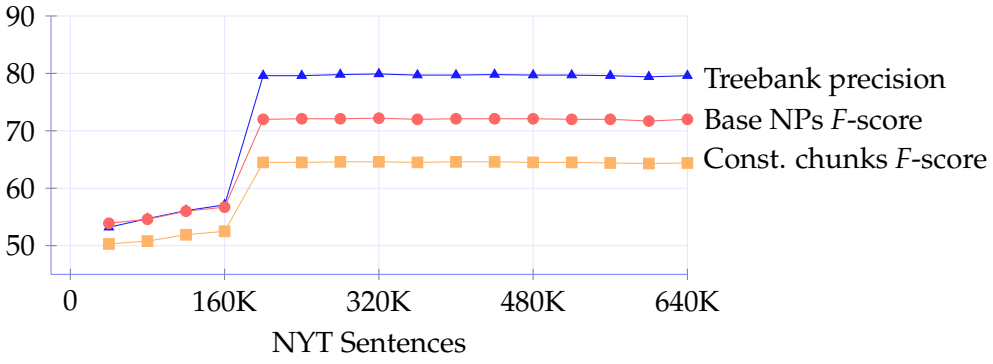
[2] `http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2003T05`
[3] `http://incubator.apache.org/opennlp/`



a) WSJ train + additional sentences from English Gigaword (NYT)



b) 2X WSJ train + additional sentences from English Gigaword (NYT)



c) Only training from English Gigaword (NYT)

Figure 3.13: Scaling experiments using New York Times data from the English Gigaword corpus. Each dot in the plots corresponds to adding 40K sentences of NYT text data, which is approximately the size of the Penn Treebank WSJ training set.

amount of data in the WSJ training dataset. Initially, scaling up the data shows positive results: improvements on all three metrics comes with the first two steps (adding 40K and 80K sentences of NYT data). However, from there results on all three metrics begins to degrade.

The overall lesson seems to be that the advantages of adding additional training data do not offset the disadvantages of using data from out of domain. While both are newstext, like any non-balanced corpus WSJ has textual properties specific to it (see Webber, 2009, for an analysis of the genres of WSJ documents).

One potential remedy for this is to boost the influence of the available in-domain material. This leads to the models being learned which are tuned for the domain, but have additional information provided by the auxiliary training material (more seen vocabulary, for one thing). Figure 3.13b provides a first stab at this approach, using a simple (and crude) method to boost the counts of the in-domain training: the in domain WSJ dataset is simply repeated in sequence. Thus, the counts for the in domain (WSJ) textual material are given twice the weight as the new material, word-for-word – though there is more new material. This turns out to have the desired effect, as the gains from adding new training material are more robust, and show improved evaluation under the three metrics through 160K sentences of additional NYT material. However, at this point, for the constituent chunk and base NP identification evaluations, results again begin to degrade.

Finally, in Figure 3.13c, the learning curve is given for experiments using *just* the NYT data as training. Again, the steps in the learning curve are 40K sentences, which is approximately the size of the WSJ training dataset. At 200K sentences, the chunker results jump from the low 50s for all three evaluation metrics to surprisingly good numbers: almost as good as the taggers trained with WSJ. The exact numbers are given in Table 3.10. These results are still below those of the model trained on WSJ, but they are not far off. Comparing the different plots, those of the experiments using both training sets seem to show model performance approaching that of the models trained using only NYT, as more NYT training data is given.

	Const. chunks			Base NPs			Treebank
	Prec.	Rec.	*F*	Prec.	Rec.	*F*	Prec.
WSJ	75.8	63.7	69.2	76.6	76.6	76.6	82.8
WSJ + 80k NYT*	76.5	63.6	69.5	77.7	76.9	77.3	84.0
WSJ + 640k NYT	72.4	58.4	64.7	73.8	70.9	72.3	81.7
2X WSJ	74.9	63.7	68.8	75.6	76.5	76.1	81.6
2X WSJ + 160K NYT*	76.1	64.3	69.7	77.0	77.4	77.2	83.3
2X WSJ + 640K NYT	72.8	58.9	65.1	74.3	71.5	72.8	81.9
560K NYT	71.8	58.4	64.4	73.2	70.8	72.0	79.6

Table 3.10: Results of the experiments using auxiliary training data from the English Gigaword (NYT) corpus. Best results in the learning curves are indicated with a star*.

POS Sequence	# of errors
TO VB	673
NNP NNP	450
MD VB	407
DT JJ	368
DT NN	280

Table 3.11: Top 5 POS sequences of the false positives predicted by the HMM.

3.3.4 Comparing the HMM and PRLG: Case Study

What distinguishes the PRLG from the HMM in these results? To outline some of the factors differentiating the HMM and PRLG, I focus on NP identification in WSJ as a case study.

The PRLG has higher precision than the HMM, while the two models are closer in recall. Comparing the predictions directly, the two models often have the same correct predictions and often miss the same gold standard constituents. The improved results of the PRLG are based mostly on the fewer overall brackets predicted, and thus fewer false positives: for WSJ the PRLG incorrectly predicts 2241 NP constituents compared to 6949 for the HMM. Table 3.11 illustrates the top 5 POS sequences of the false positives predicted by the HMM.[4] (Keep in mind gold standard POS are used *only* for post-experiment results analysis—the model itself does not have access to them.) By contrast, the sequence representing the largest

[4]For the Penn Treebank tagset, see Marcus et al. (1993).

class of errors of the PRLG is DT NN, with 165 errors – this sequence represents the largest class of predictions for both models.

Two of the top classes of errors, MD VB and TO VB, represent verb phrase constituents, which are often predicted by the HMM chunker, but not by the PRLG. The class represented by NNP NNP corresponds with the tendency of the HMM chunker to split long proper names: for example, it systematically splits *new york stock exchange* into two chunks, (*new york*) (*stock exchange*), whereas the PRLG chunker predicts a single four-word chunk.

The most interesting class is DT JJ, which represents the difficulty the HMM chunker has at distinguishing determiner-adjective from determiner-noun pairs. The PRLG chunker systematically gets DT JJ NN trigrams as chunks. The greater context provided by right branching rules allows the model to explicitly estimate separate probabilities for $\Pr(\textsc{i} \rightarrow \textit{recent}\ \textsc{i})$ versus $\Pr(\textsc{i} \rightarrow \textit{recent}\ \textsc{o})$. That is, *recent* within a chunk versus ending a chunk. Bigrams like *the acquisition* allow the model to learn rules $\Pr(\textsc{b} \rightarrow \textit{the}\ \textsc{i})$ and $\Pr(\textsc{i} \rightarrow \textit{acquisition}\ \textsc{o})$. So, the PRLG is better able to correctly pick out the trigram chunk (*the recent acquisition*).

3.3.5 Model analysis

Generalizing on this discussion, the following section reports investigations into the generalizations the sequence models seem to be learning. This is done by inspecting the probability distributions learned directly.

Hidden Markov models

To investigate the HMMs trained by EM, we can inspect the emission probability distributions directly: the probabilities $\Pr(w|q)$ of a word w being emitted by a state q. These are displayed in Figure 3.14.

English/WSJ For the WSJ dataset, the models learn a strong affinity between the B tag and determiners, witness the strong emission probabilities for *the*, *a*, *an* and *its* (a possessive determiner). That *mr.* has a high emission probability from the B state makes sense, and provides a key into the model's ability to identify (rare) names as chunks: from *mr. humperdinck* the model will associate *humperdinck* with the I tag, and so in another sentence correctly identify *engelbert humperdinck* as a B–I

B	$\Pr(w\|\text{B})$	I	$\Pr(w\|\text{I})$	O	$\Pr(w\|\text{O})$
the	21.0	%	1.8	of	5.8
a	8.7	million	1.6	and	4.0
to	6.5	be	1.3	in	3.7
's	2.8	company	0.9	that	2.2
in	1.9	year	0.8	to	2.1
mr.	1.8	market	0.7	for	2.0
its	1.6	billion	0.6	is	2.0
of	1.4	share	0.5	it	1.7
an	1.4	new	0.5	said	1.7
and	1.4	than	0.5	on	1.5
$H(W\|\text{B}) = 7.84$		$H(W\|\text{I}) = 11.86$		$H(W\|\text{O}) = 9.55$	

a) WSJ

B		$\Pr(w\|\text{B})$	I		$\Pr(w\|\text{I})$	O		$\Pr(w\|\text{O})$
der	*the*	13.0	uhr	*o'clock*	0.8	in	*in*	3.4
die	*the*	12.2	juni	*June*	0.6	und	*and*	2.7
den	*the*	4.4	jahren	*years*	0.4	mit	*with*	1.7
und	*and*	3.3	prozent	*percent*	0.4	für	*for*	1.6
im	*in*	3.2	mark	*currency*	0.3	auf	*on*	1.5
das	*the*	2.9	stadt	*city*	0.3	zu	*to*	1.4
des	*the*	2.7	000		0.3	von	*of*	1.3
dem	*the*	2.4	millionen	*millions*	0.3	sich	*oneself*	1.3
eine	*a*	2.1	jahre	*year*	0.3	ist	*is*	1.3
ein	*a*	2.0	frankfurter	*Frankfurt*	0.3	nicht	*not*	1.2
$H(W\|\text{B}) = 8.52$			$H(W\|\text{I}) = 13.92$			$H(W\|\text{O}) = 11.07$		

b) Negra

B		$\Pr(w\|\text{B})$	I		$\Pr(w\|\text{I})$	O		$\Pr(w\|\text{O})$
的	de, *of*	14.3	的	de	3.9	在	*at, in*	3.4
一	*one*	3.1	了	(perf. asp.)	2.2	是	*is*	2.4
和	*and*	1.1	个	*ge* (measure)	1.5	中国	*China*	1.4
两	*two*	0.9	年	*year*	1.3	也	*also*	1.2
这	*this*	0.8	说	*say*	1.0	不	*no*	1.2
有	*have*	0.8	中	*middle*	0.9	对	*pair*	1.1
经济	*economy*	0.7	上	*on, above*	0.9	和	*and*	1.0
各	*each*	0.7	人	*person*	0.7	的	de	1.0
全	*all*	0.7	大	*big*	0.7	将	fut. tns.	1.0
不	*no*	0.6	国	*country*	0.6	有	*have*	1.0
$H(W\|\text{B}) = 10.50$			$H(W\|\text{I}) = 11.45$			$H(W\|\text{O}) = 10.81$		

c) CTB

Figure 3.14: The 10 most probable words for each state under the emission probabilities learned by the HMM.

tag sequence, *i.e.* a chunk. The word type *'s* has as high probability under the B label, and this corresponds to a number of the errors made by this model, incorrect chunk predictions like

(the economy) ('s temperature)

thursday ('s report)

friday ('s session)

Similarly, the high probability of the word type *and* under the B label leads to a number of errors, such as

housing (and inflation)

(industrial production) (and capacity utilization)

genius (and energy)

Finally, prepositions *to*, *in* and *of* have a high probability under the label B, meaning a number of smaller prepositional phrases will be identified as chunks.

The I label emits many content terms – nouns and adjectives – with high probability. This being the Wall Street Journal, many have to do with quantitative market and company information (%, *million*, *company*, *billion*, *share*). *new* has a high emission probability under I; inspecting the chunker output, this seems to contribute to a number of correct 3+ word chunks:

(a new agreement)

(the new agreement)

(a new bid)

(the new financing structure)

Figure 3.14a indicates that the term *than* is emitted by the I label with high probability. Inspection of chunker output suggests that this is due in part to common chunk predictions for the expressions (*less than*) and (*more than*), which are linguistically defensible chunks, albeit false positives compared to Penn Treebank annotations – though *rather than* frequently is annotated as a constituent. Finally, the high frequency of the term *be* under the I label corresponds to frequently predicted chunks

(*to be*), (*will be*), (*may be*), (*should be*). Again, these are linguistically plausible constituents, though in most cases are false positives. Some predictions for I emitting *be* are less defensible: for example (*n't be*) or (*to be*) *sure*.

The O label has high probability emissions for function words like prepositions (*of*, *in*, *to*, *for*, *on*), the declined copula *is*, the conjunction *and*, the high frequency verb *said* and the pronoun *it*. The term *that* is flexible in English, as it can be a demonstrative pronoun (*that was it*), a demonstrative adjective or determiner (*that win put them in the championship*), a complementizer (*he said that they won*), among others. Inspecting model output, it seems that most instances of *that* as a demonstrative adjective were correctly placed at the beginning of a chunk.[5]

Just looking at these top-ten emissions and the emission probabilities, the distribution of terms emitted by the B label seems much more skewed, or spiky, than that for the I label which seems to be a much flatter distribution over nouns and content terms. Indeed it is: one way of quantifying this is by observing the information-theoretic *entropy* of these distributions. The entropy $H(X)$ of a random variable X is a measure of the uncertainty of X; specifically,

$$H(X) = \sum_x \Pr(x) \log_2 \Pr(x)$$

for all x, possible values of X. A common intuition is: take given $x \sim X$ to be a *message* emitted by X, $H(X)$ is the number of bits, on average, required to encode x (hence $\log_2$). For a skewed probability distribution, a shorter encoding can be used for the high-frequency messages (Manning and Schütze, 1999, have a nice tutorial on this). The entropy of the word types emitted by each of the chunker labels ($W|$B, $W|$I, $W|$O) are given in Figure 3.14; and, it turns out that the B and O labels, which seem to have spikier distributions favoring function words disproportionately higher, do in fact have lower entropy than the I label: 7.84 bits and 9.55 bits, respectively, versus 11.86 bits. Curiously, many of the top 10 emissions of the B and O labels are orthographically shorter than the top ten emissions of the I label.

[5]This actually speaks to a nice property of this model, that in at least some cases this model will assign different labels to terms, correctly, in different contexts. In unsupervised part-of-speech identification, from Brown et al. (1992) to Blunsom and Cohn (2011), a heuristic constraint compelling only one tag to be assigned to all tokens of a given word type has frustratingly led to better results, in spite of the fact that linguistically this is not a legitimate constraint. Future work may combine the insights of an HMM-based unsupervised chunker with POS tagger learning to relax this constraint.

German/Negra The HMM trained from the Negra data shows very similar patterns. The B label emits determiners with high probability. The I label emits nouns and content terms, like time and temporal locations (*uhr*, *jahre*, *jahren*, *juni*), currency and quantities (*prozent*, *mark*, *000*, *millionen*) and places (*stadt*, *frankfurter*). The O label emits others, including prepositions (*in*, *mit*, *für*, *auf*, *zu*, *von*) the copula *ist*, and, here, other functional words (*sich*, *und*). The term *und* is also emitted by the B with high probability, and inspection of model output found that many (if not most) predicted chunks beginning with *und* were false positives. Again, looking at the entropy of the distributions associated with the three labels, a similar pattern to the HMMs trained on WSJ emerges, that the B has the least entropy at 8.52 bits, and the I has the most at 13.92 bits; the O has 11.07.

Chinese/CTB The models' parameters learned from the Chinese treebank, and presented in Figure 3.14, are a little harder to interpret, and may indicate that concise models for unsupervised chunking of Chinese are difficult to learn in an unsupervised fashion. To begin with the 的 (de) term is emitted with high probability by all three labels, and it is the highest probability term under B and I. This is a very common character, and its most common grammatical purpose is a possessive marker. The grammatical pattern marking a possessive relationship in Chinese is POSSESSOR 的 POSSESSED, similar to English *'s*. Inspection of model output suggests that – similar to the mistakes the HMM model was predicting chunks beginning with *'s* on WSJ, many of the chunks predicted beginning with 的 are false positives. For instance, in

中国	的	经济	发展
China	*de*	economy	development

the correct bracketing is

(中国 的) (经济 发展)

but model output is

中国 (的 经济 发展)

Similar problems arise in other constructions, for instance attributive adjectives are commonly indicated in Chinese by ADJECTIVE 的 NOUN, and in these cases the

HMM chunker frequently – incorrectly – identified the 的 NOUN sequence as a chunk. In the cases where 的 was associated with the I, it is more often the correct prediction.

Other emissions of the B label are less problematic: for instance determiners such as 一 (*yī*, 'one'), 两 (*liǎng*, 'two'), 各 (*gè*, 'each'), 全 (*quán*, 'all'). The noun 经济 (*jīngjì*, 'economy') is frequently chunked with 的, correctly, to roughly mean *economic* (经济 的). The term 不 (*bù*, 'not') has – like *not* in English – both an adverbial function (*not good*) and a function as sentential negation. The HMM actually learns these different roles reasonably well, though not perfectly, associating 不 with the B label for adverbial usage, as in 不 够高 (*not high enough*) and 不 公正 (*not fair*); and associating it with the O in sentential negation contexts. The term 和 (*hé*, 'and') has high probability under the B, corresponding with frequent – incorrect – predictions of chunks beginning with this term; though, the HMM makes similar generalizations and predictions with *and* in WSJ and *und* in Negra.

The terms related to the I contains more of a mixture of function words and content words than the models trained on WSJ and Negra. For instance, the term 人 (*ren*, 'person') frequently functions with a country term or other modifier, for example 中国 人 *Chinese* (中国 = 'China'). The 了 (*le*) term is a perfective marker, usually appearing immediately after a verb; the HMM frequently predicts VERB 了 pairs as chunks, as in 主持 了 (*presided over*), 听取 了 (*heard*), though these are not annotated as constituents in the CTB. The model did correctly identify the sequence

发	了	言
issue	*le*	word

meaning *made a statement*, as a constituent chunk. It is possibly due to this mixture of high frequency function words and content words that the entropy of the distribution associated with the I label, in CTB, is not as distinctly higher than that of B or O as in the other datasets.

Probabilistic right linear grammars

Likewise, the 15 highest probability right-linear rules generated by the PRLG models are displayed in Figures 3.15 (WSJ), 3.16 (Negra) and 3.17 (CTB).

B		$\Pr(\text{B} \to w\,q)$	I		$\Pr(\text{I} \to w\,q)$	O		$\Pr(\text{O} \to w\,q)$
B →	the I	28.2	I →	's I	2.6	O →	of B	3.8
B →	a I	11.7	I →	and I	1.3	O →	to O	3.6
B →	mr. I	2.4	I →	% O	1.1	O →	in B	2.5
B →	its I	2.2	I →	million O	0.6	O →	and O	1.7
B →	an I	1.9	I →	new I	0.5	O →	to B	1.7
B →	his I	1.0	I →	million STOP	0.5	O →	of O	1.6
B →	this I	1.0	I →	company O	0.5	O →	in O	1.5
B →	their I	1.0	I →	year O	0.4	O →	and B	1.4
B →	some I	0.7	I →	& I	0.4	O →	for B	1.3
B →	new I	0.6	I →	million I	0.4	O →	it O	1.3
B →	other I	0.5	I →	% STOP	0.3	O →	is O	1.3
B →	one I	0.5	I →	share STOP	0.3	O →	that O	1.2
B →	last I	0.4	I →	stock I	0.3	O →	on B	0.9
B →	that I	0.4	I →	year STOP	0.3	O →	he O	0.8
B →	any I	0.4	I →	market O	0.3	O →	from B	0.8
	$H(W,Q\|\text{B}) = 7.88$			$H(W,Q\|\text{I}) = 12.94$			$H(W,Q\|\text{O}) = 10.52$	

Figure 3.15: 15 highest probability PRLG rules per state: **WSJ**

B			$\Pr(\text{B} \to w\,q)$	I			$\Pr(\text{I} \to w\,q)$	O			$\Pr(\text{O} \to w\,q)$
B →	der I	*the*	12.4	I →	mark O	*(currency)*	0.3	O →	in B	*in*	2.5
B →	die I	*the*	11.7	I →	uhr O	*o'clock*	0.3	O →	zu O	*to*	1.2
B →	den I	*the*	4.2	I →	jahren O	*years*	0.2	O →	und O	*and*	1.2
B →	und I	*and*	3.9	I →	bis B	*to*	0.2	O →	auf B	*on*	1.2
B →	im I	*in*	3.0	I →	juni O	*June*	0.2	O →	und B		1.2
B →	das I	*the*	2.7	I →	juni STOP		0.2	O →	mit B	*with*	1.1
B →	des I	*the*	2.6	I →	jahr O	*year*	0.2	O →	für B	*for*	1.1
B →	dem I	*the*	2.4	I →	jahre O	*years*	0.2	O →	nicht O	*nicht*	1.1
B →	ein I	*a*	2.1	I →	000 I		0.2	O →	es O	*is*	0.9
B →	eine I	*a*	2.0	I →	stadt O	*city*	0.2	O →	sich O	*oneself*	0.8
B →	von I	*of*	1.9	I →	frankfurter I	*Frankfurt*	0.2	O →	ist O	*is*	0.7
B →	in I	*in*	1.9	I →	neuen I	*new*	0.1	O →	von B	*of*	0.7
B →	am I	*at the*	1.5	I →	millionen I	*millions*	0.1	O →	sie O	*they*	0.7
B →	mit I	*with*	1.0	I →	ersten I	*first*	0.1	O →	an B	*to*	0.6
B →	zum I	*to*	1.0	I →	uhr B	*o'clock*	0.1	O →	auch B	*also*	0.6
	$H(W,Q\|\text{B}) = 8.16$				$H(W,Q\|\text{I}) = 15.78$				$H(W,Q\|\text{O}) = 12.73$		

Figure 3.16: 15 highest probability PRLG rules per state: **Negra**.

B		$\Pr(\text{B} \to w\ q)$	I		$\Pr(\text{I} \to w\ q)$	O		$\Pr(\text{O} \to w\ q)$
B → 一 I	*one*	2.9	I → 的 I	de, *of*	10.6	O → 在 B	*at, in*	2.9
B → 中国 I	*China*	2.3	I → 个 I	ge *(measure)*	0.9	O → 不 O	*no*	1.6
B → 了 I	perf. asp.	2.3	I → 的 B	de	0.8	O → 是 B	*is*	1.4
B → 这 I	*this*	1.5	I → 和 I	*and*	0.7	O → 也 O	*also*	1.3
B → 台湾 I	*Taiwan*	1.5	I → 年 I	*year*	0.7	O → 是 O	*is*	1.2
B → 两 I	*two*	0.9	I → 一 I	*one*	0.7	O → 有 B	*have*	1.0
B → 全 I	*all*	0.8	I → 大 I	*big*	0.7	O → 就 O	*at once*	0.8
B → 香港 I	*Hong Kong*	0.8	I → 经济 I	*economy*	0.7	O → 说 STOP	*say*	0.8
B → 记者 I	*reporter*	0.7	I → 和 B	*and*	0.5	O → 他 O	*he*	0.7
B → 国际 I	*international*	0.6	I → 等 I	*rank*	0.5	O → 对 B	*pair*	0.7
B → 新 I	*new*	0.6	I → 国 I	*country*	0.4	O → 以 B	*use*	0.7
B → 美国 I	*USA*	0.6	I → 投资 I	*investment*	0.4	O → 在 O	*at, in*	0.6
B → 各 I	*each*	0.6	I → 企业 I	*company*	0.3	O → 要 O	*must, ask*	0.6
B → 世界 I	*world*	0.5	I → 发展 I	*development*	0.3	O → 为 B	*act as; because*	0.6
B → 三 I	*three*	0.5	I → 与 I	yu *(part.)*	0.3	O → 都 O	*all*	0.6
$H(W, Q\|\text{B}) = 11.28$			$H(W, Q\|\text{I}) = 12.19$			$H(W, Q\|\text{O}) = 12.01$		

Figure 3.17: 15 highest probability PRLG rules per state: **CTB**.

WSJ These tables of rule probabilities help quantify what the PRLG is learning, versus the HMM. There are familiar patterns with the terms associated with the B label: determiners, possessive pronouns, *mr.*; and also interestingly the term *last*, as it appears in a number of temporal location NPs:

(last month)

(last year 's unusually high level)

(last week)

(last night)

However, with the I label, the PRLG is able to make generalizations that the HMM does not. Consider the term *'s*. A common pattern for this term is that it immediately precedes nouns and adjectives, which, due to the strong determiner-adjective-noun pattern, the model associates with the I label. The *'s* term does not occur at the beginnings of sentences very often, though due to the fact it consistently appears before adjectives and nouns, the HMM learns to associate it with the B label. The more granular rules of the PRLG seem to enable it to make a better generalization: the term *'s* commonly occurs *within NPs*, yet it also commonly occurs *before nouns and adjectives*. This is summarized by the rule I → *'s* I, which, in fact, has the

highest probability under I in Figure 3.15.

More generally, these tables show where the PRLG is able to learn a little more grammar than the HMM. For instance, with the O label, it learns to associate certain terms – prepositions, largely, immediately before NP-initial terms, *viz.* determiners. Specifically, it learns high probabilities for rules:

O → *of* B

O → *to* B

O → *for* B

O → *on* B

O → *from* B

For certain terms, such as *he*, it learns the rule O → *he* O, indicating it occurs before non-NPs frequently. Similarly, with the rules associated with the I label, the PRLG learns certain terms which tend to end NPs, via rules like

I → % O

I → % STOP

I → *year* O

I → *year* STOP

I → *share* O

as well as terms which tend to be part of continued noun phrases, such as *'s*, via rules:

I → *and* I

I → *new* I

I → & I

I → *stock* I

Finally, a similar pattern emerges with respect to the entropy of the probability distributions associated with the three labels as seen with the HMM emissions.

Negra Similar patterns characterize the PRLG rules learned from the Negra corpus, and the patterns here are similar to the emission probabilities learned by the HMM. The rules associated with the I label distinguish NP-concluding terms (*mark*, *uhr*, *juni*, *jahren*, *jahre*, *jahre*, *stadt*) from NP-continuing terms (*000*, *frankfurter*, *neuen*, *millionen*, *ersten*). Likewise, with the O, the rules distinguish terms – prepositions, mostly – which occur before NP chunks (*in*, *auf*, *mit*, *für*, *von*, *auch*) from those which occur before non-NP chunks (*zu*, *und*, *nicht*, *es*, *sich*, *ist*, *sie*).

CTB The PRLG trained from CTB shows some of the same patterns as the HMM did: many of the same function words are associated with the O, many of the same quantifiers are associated with the B. Though, probably the biggest change mirrors the difference cited above between the PRLG and the HMM trained on WSJ: the possessive marker 的 (*de*) is no longer associated with the B, and factored into two rules associated with the I label. What this indicates – the rule I $\rightarrow$ 的 I in particular – is that given a sequence POSSESSOR 的 POSSESSED, rather than chunking POSSESSOR (的 POSSESSED), as does the HMM, the PRLG tends to chunk the whole sequence: (POSSESSOR 的 POSSESSED). Unfortunately, under the constituent chunk and base NP identification evaluations, this is usually incorrect: the correct chunking is (POSSESSOR 的) (POSSESSED). This, again, suggests that even as the PRLG is capable of discovering grammatical structure, the constituent chunk and base NP evaluations are do not overlap well with PRLG predictions.

3.3.6 Limitations of the PRLG model

Hidden Markov models, in the general case, are *reversible*. That is, for training and decoding, it does not make any difference whether – algorithmically – they are run left to right or right to left. Alternatively, if the string of terms provided to an HMM as training is reversed – assuming the initial parameters are symmetric – then the resulting model will be the same, only in reverse. This does not apply to HMMs with the BIO tagset, as it has been presented so far, since the initial parameters of the HMM are not symmetric, and are constrained never to be. In particular, an I can follow an I, but this is not the case with a B.

This has consequences for natural language processing applications. Certain languages, such as Japanese, have 'reversed' features vis-à-vis English gram-

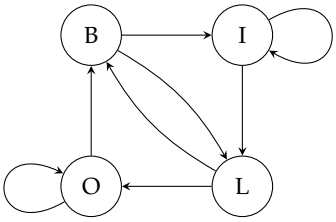


Figure 3.18: BILO tag transition diagram. Note that STOP and O pattern together, so do not need to both be portrayed in this diagram.

matical structures. Japanese has post-positions rather than prepositions; word-final particles rather than phrase-initial determiners;[6] and sentential phrases are generally subject-object-verb, compared to subject-verb-object in English.[7]

As a test of the robustness of these structure learning models, consider an experiment wherein both training data, evaluation data and corresponding gold-standard annotations for evaluation are all reversed. Klein and Manning's CCM is completely symmetric, its evaluation would be identical to the normal (non-reversed case). Seginer's CCL is not completely symmetric, but Seginer (2007b) ran these 'reversed' experiments and found CCL performed nearly as well. This is not the case for some of the sequence models. Table 3.12 has evaluation results of these 'reversed' experiments on the proposed models.

Curiously, the HMM does slightly better for WSJ and Negra when run in reverse. But the PRLG does much worse. One obvious first place to look is at the BIO tagset used. These tags, and the constraints that define them, are intrinsically asymmetrical: chunks are characterized by a tag sequences like BI, BII, *etc.*, where there is a distinguished start state but no distinguished end state. A solution is to define a model which has states indicating both the beginning and end of constituents. *BILO* is just such as tagset: BIO plus a new state L for the last word of a constituent. As before, initial state-transitions are used to constrain the possible tag sequences. The corresponding state-transition diagram is given in Figure 3.18.

However, as Table 3.13 shows, this is not enough: as before the PRLG model predictions are different when run in reverse. And, in almost all cases, as before

[6]Granted, comparing determiners to particles is not really accurate.

[7]Though, Japanese is much more flexible in ordering of subject, object and verb than English.

		WSJ			Negra			CTB		
		Prec.	Rec.	*F*	Prec.	Rec.	*F*	Prec.	Rec.	*F*
HMM	LR	52.8	61.5	56.8	33.9	39.5	36.5	36.6	41.4	38.9
	RL	54.2	61.7	57.7	37.1	43.7	40.1	30.4	32.3	31.3
PRLG	LR	76.1	63.7	69.3	39.6	51.1	44.6	22.7	18.5	20.4
	RL	28.7	30.3	29.5	28.9	46.2	35.6	21.2	23.7	22.4

a) Constituent chunk evaluation

		WSJ			Negra			CTB		
		Prec.	Rec.	*F*	Prec.	Rec.	*F*	Prec.	Rec.	*F*
HMM	LR	47.2	65.3	54.8	23.3	46.8	31.1	17.4	31.4	22.4
	RL	48.3	65.5	55.6	25.4	51.8	34.1	15.8	26.6	19.8
PRLG	LR	76.9	76.5	76.7	24.5	54.6	33.8	21.8	28.3	24.6
	RL	20.0	25.1	22.3	13.2	36.4	19.3	8.9	15.9	11.4

b) Base NPs evaluation

		WSJ	Negra	CTB
HMM	LR	54.5	33.9	42.8
	RL	56.3	37.4	38.4
PRLG	LR	83.1	40.2	44.8
	RL	48.1	32.2	35.0

c) Precision on all constituents

Table 3.12: Effects of reversing training and evaluation datasets on sequence model performance. LR are results on non-reversed datasets, quoting from before; RL are results on reversed datasets.

the PRLG results are worse.

Even with the asymmetric BIO coding, the HMM's performance at the chunking and NPs tasks is not so different when the dataset is reversed. With the BILO coding, the model is reversible, as expected. However the PRLG performance is significantly downgraded. Firstly, note that the PRLG is not symmetric, as an HMM is. In fact, the PRLG is one of two generalizations of an HMM, the other being a probabilistic *left-linear* grammar. The difference has to do with how the joint arc-probability between to states is calculated, whether the first emission is factored in or the second. This difference is illustrated in Figure 3.19.

PLLG rules are not as natural a fit for a language like English as PRLG. The

		WSJ			Negra			CTB		
		Prec.	Rec.	*F*	Prec.	Rec.	*F*	Prec.	Rec.	*F*
HMM	both	26.0	29.8	27.8	48.6	59.3	53.4	17.6	11.9	14.2
PRLG	LR	72.8	62.7	67.4	10.8	11.5	11.1	19.8	14.4	16.7
	RL	21.6	24.8	23.1	20.3	27.3	23.3	19.7	20.7	20.2

a) Constituent chunk evaluation

		WSJ			Negra			CTB		
		Prec.	Rec.	*F*	Prec.	Rec.	*F*	Prec.	Rec.	*F*
HMM	both	17.9	24.3	20.6	25.5	53.9	34.7	14.4	15.5	14.9
PRLG	LR	72.7	74.5	73.6	7.5	13.7	9.7	22.6	26.1	24.2
	RL	10.8	14.8	12.5	7.3	16.8	10.1	7.0	11.8	8.8

b) Base NPs evaluation

		WSJ	Negra	CTB
HMM	both	45.8	53.2	52.8
PRLG	LR	80.7	23.0	48.9
	RL	42.0	28.1	39.1

c) Precision on all constituents

Table 3.13: Unsupervised chunking results with BILO model. As before, LR means the model is run on the original dataset, RL means dataset is reversed.

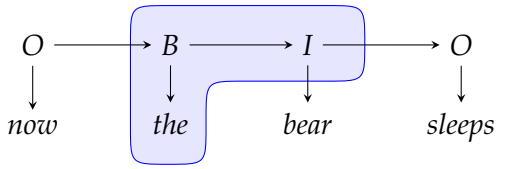
PRLG rule

$$\Pr(B \rightarrow \textit{the}\ I) \tag{3.8}$$

captures the intuition that the determiner *the* often begins a phrase. As English has more phrase-beginning function words (determiners, prepositions, auxiliary verbs, *etc.*), rules like this will be able to capture this generalization. The affinity for the B tag and terms like determiners is hard to characterize in a PLLG since the probabilities will be split between three different rules:[8]

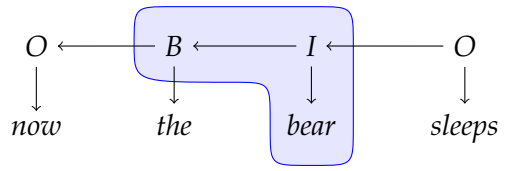
$$\Pr(O \rightarrow B\ \textit{the})$$
$$\Pr(I \rightarrow B\ \textit{the})$$
$$\Pr(\textit{STOP} \rightarrow B\ \textit{the})$$

[8]Depending on the tagset.



$\Pr(B \rightarrow \textit{the } I)$

a) Probabilistic right-linear grammar (PRLG)



$\Pr(I \rightarrow B \textit{ bear})$

b) Probabilistic left-linear grammar (PLLG)

Figure 3.19: PRLG and probabilistic left-linear grammar (PLLG) joint predictions.

When using a symmetric chunking tagset, like BILO, running the reverse experiments, as in Table 3.13, is equivalent to learning an unsupervised chunking using a PLLG on the non-reversed dataset. With an asymmetric tagset, it is not equivalent to a PLLG, but poses similar problems for the model, if the goal is to learn generalizations like (3.8).

3.3.7 Phrasal punctuation

Table 3.14 provides comparison of the sequence models' performance on the constituent chunking task without using phrasal punctuation in training and evaluation. (To be clear, even in experiments where models have access to phrasal punctuation at test time, punctuation is removed from the text before final evaluation.) The table shows absolute improvement (+) or decline (−) in precision and recall when phrasal punctuation is removed from the data.

Phrasal punctuation clearly helps the chunking models, especially on WSJ where the gains from using punctuation in this way are quite large. For Negra and CTB, though it is not as clear: the gains from using phrasal punctuation to chunk these datasets are much smaller or non-existent. One major exception is base NP identification, using the PRLG model. Phrasal punctuation also helps with the

		WSJ			Negra			CTB		
		Prec.	Rec.	F	Prec.	Rec.	F	Prec.	Rec.	F
HMM	W/ Punctuation	52.8	61.5	56.8	33.9	39.5	36.5	36.6	41.4	38.9
	No Punctuation	47.3	52.0	49.5	33.6	38.5	35.9	37.7	46.3	41.6
	Difference	−5.5	−9.5	−7.3	−0.3	−1.0	−0.6	+1.1	+4.9	+2.7
PRLG	W/ Punctuation	76.1	63.7	69.3	39.6	51.1	44.6	22.7	18.5	20.4
	No Punctuation	72.8	61.4	66.6	37.5	48.8	42.4	15.5	19.2	17.1
	Difference	−3.3	−2.3	−2.7	−2.1	−2.3	−2.2	−7.2	+0.7	−3.3

a) Constituent chunking evaluation

		WSJ			Negra			CTB		
		Prec.	Rec.	F	Prec.	Rec.	F	Prec.	Rec.	F
HMM	W/ Punctuation	47.2	65.3	54.8	23.3	46.8	31.1	17.4	31.4	22.4
	No Punctuation	44.3	57.9	50.2	23.6	46.8	31.4	15.8	31.0	21.0
	Difference	−2.9	−7.4	−4.6	+0.3	−0.0	+0.3	−1.6	−0.4	−1.4
PRLG	W/ Punctuation	76.9	76.5	76.7	24.5	54.6	33.8	21.8	28.3	24.6
	No Punctuation	73.6	73.8	73.7	23.2	52.2	32.1	10.0	19.7	13.2
	Difference	−3.3	−2.6	−3.0	−1.3	−2.4	−1.7	−11.8	−8.6	−11.4

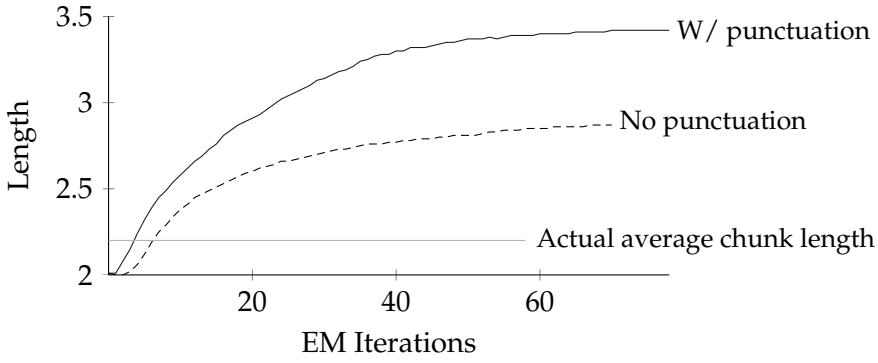
b) Base NP evaluation

Table 3.14: Evaluation of chunking models with and without phrasal punctuation. Without phrasal punctuation, only sentence boundaries are treated as boundary symbols (*i.e.* have tag STOP).

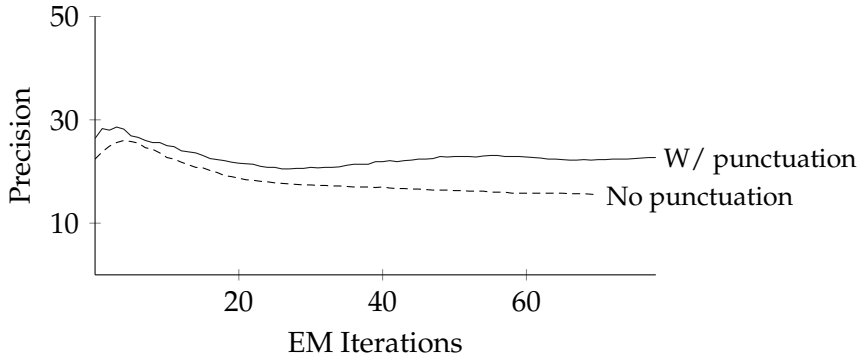
constituent chunking task for CTB for the PRLG model, but on this task the PRLG under-performs the HMM by a large margin.

3.3.8 CTB chunking

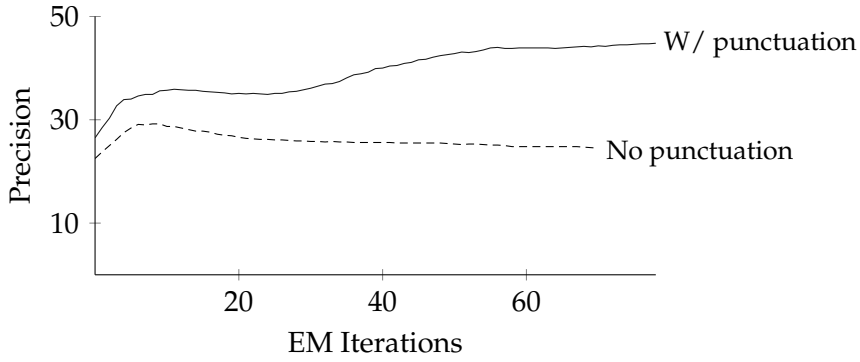
For the CTB dataset, in contrast to many of the other experiments, the PRLG is actually the worst performing model at the constituent chunking task; also, this model has the largest drop in performance when phrasal punctuation is dropped from input. To investigate, the performance of the PRLG chunkers on the development dataset, over iterations of EM, is plotted in Figure 3.20. First of all, Figure 3.20a reveals the average length of the constituents predicted by the PRLG model increases over the course of EM. However, the average constituent chunk length is 2.22. So, the PRLG chunker is predicting constituents that are longer than the ones



a) Average chunk length



b) Constituent chunking precision



c) All treebank constituents precision

Figure 3.20: Behavior of the PRLG model on CTB over the course of EM.

targeted in the constituent chunking task: regardless of whether they are legitimate constituents or not, often they will likely be counted as false positives in this evaluation. This is confirmed by observing the constituent chunking precision in Figure 3.20b, which peaks when the average predicted constituent length is about the same the actual average length of those in the evaluation. The question, then, is whether the longer chunks predicted correspond to actual constituents or not. Figure 3.20c shows that the PRLG, when constrained by phrasal punctuation, does continue to improve its constituent prediction accuracy over the course of EM. These correctly predicted constituents are not counted as such in the constituent chunking or base NP evaluations, but they factor directly into improved accuracy when this model is part of a full parsing cascade.

3.3.9 Computational analysis

The learning procedure of EM for HMMs has time complexity $O(NT^2)$ for dataset size N – the number of words in the training set, in this application – and the number of tags T. This does not change with PRLGs: what does change is the number of parameters required for a model. For an HMM, $T^2 + TV$ parameters are required, for V the size of the vocabulary of terms, whereas for a PRLG T^2V parameters are required. The point is that, for a fixed number of tags T, the time required to train an HMM or PRLG scales linearly with the length of the dataset. In other words, sticking to a standard (and small) tagset like BIO or BILO, the T^2 term is essentially a constant factor.

One factor of the sequence models' time complexity that is difficult to predict is the number of iterations required to converge. In other words, the full training time complexity is $O(NT^2I)$ for the number of iterations I. However, usually, letting EM run longer leads to better results, for which reason EM is run to convergence in these results.

3.4 Summary

This chapter presented a new perspective on the problem of unsupervised constituent parsing. The models proposed in this chapter focus on identifying non-overlapping constituents, that is, the task of *unsupervised partial parsing*. To be

more precise in articulating this task, direct evaluations for it are defined, based on gold-standard constituent treebank annotations: *constituent chunk identification* – the identification of lowest branching constituents – and *base NP identification* – the identification of non-nested NPs. Evaluation uses local constituents extracted from a state-of-the-art unsupervised parser as a benchmark, but show that chunkers using standard probabilistic sequence taggers – hidden Markov models – often outperform this benchmark on the tasks as defined. A variation of HMMs, probabilistic right linear grammars, often perform better still, especially for base noun phrase identification in English. However, careful analysis suggests that this model class is somewhat more sensitive to assumptions about linguistic structure than the HMM. In the following chapter, unsupervised partial parsing is applied in a cascaded fashion to produce full parse trees.

Chapter 4

Unsupervised parsing with a cascade

Sequence models work well for chunking, or predicting a single level of constituent structure, as discussed in the previous chapter. They have high precision on treebank constituents, and good precision and recall on local (lowest) constituents, and base NPs.

The original and primary motivation for unsupervised partial parsing is as a first step in a general unsupervised parsing framework. By identifying local constituents with greater accuracy than general unsupervised parsing methods are capable of, general parsing strategies can build on these local predictions for overall better parsing results.

In this chapter, a variation of this general strategy is proposed and evaluated. Rather than combining unsupervised partial parsing with a different parsing strategy, this chapter presents a *cascaded parsing* strategy to iteratively produce longer constituents from a previous round of chunking, ultimately producing full unlabeled constituent trees. This strategy turns out to work well: for evaluation using WSJ, Negra and CTB, this strategy outperforms the CCL parser of Seginer (2007a), the current state-of-the-art for unsupervised constituent parsing. In many cases, this strategy is competitive with or superior to the CCM parser of Klein and Manning (2002; 2004; see also discussion on Chapter 2), even though the latter is provided with gold-standard POS tags as input.

In this chapter the strategy for parsing using cascaded chunking models is

outlined and motivated, then empirical evaluation is provided, in terms of comparison of model output to gold-standard treebank annotations to benchmarks. Results analysis and model analysis follows. Some content from this chapter is reported in Ponvert et al. (2011).

4.1 Motivating cascaded parsing

Chapter 3 demonstrated that a direct approach to predicting local constituents can be more accurate than general a state-of-the-art general unsupervised parsing system, at least for local constituents. The hypothesis behind the direct approach is that constituents can be identified by models based on statistics for word distributions with respect to phrasal boundaries. The concrete implementation of this hypothesis is a treatment of the chunking as a tagging problem, with a specific set of tags and constraints over them. This allows statistical sequence models (HMMs and PRLGs) to learn word sequence patterns which reliably correspond to local constituents. Moving beyond local constituents, can this hypothesis apply to the general – hierarchical – constituent parsing problem?

In this chapter I argue that it can, and present a class of models for unsupervised constituent parsing which do so, achieving state-of-the-art results for parsing all three of the datasets considered. This class of models literally builds constituent structure up from the output of the unsupervised chunkers by applying a *cascaded parsing* (or *cascade of chunkers*) strategy. The idea is: first an unsupervised chunker identifies local constituents in text, then a subsequent chunker in the *cascade* identifies new constituents on top of the words and chunks already identified. This process is iterated until a full constituent tree structure has been formed.

The motivation for this approach very much follows the motivations behind the cascaded chunking strategy from Chapter 3. Noting that the latter identifies base noun phrases with good accuracy (especially for languages like German and English), the expectation for the cascaded parsing strategy is that other local constituents, at the next level of constituent structure, can be identified by just the same method of generalizing on phrasal boundaries. These constituents at the next level of constituent structure include things like prepositional phrases and recursive noun phrases (NPs with embedded NPs). Specifically, the expectation is that the very same classes statistical sequence models used to identify local constituents

can also learn to identify non-local constituents. The cascaded parsing strategy fulfills this expectation, and moreover subsequent level in the cascade constribute further to overall more accurate constituent trees, in direct evaluation. This strategy and these results are described in the remainder of this chapter.

4.2 An architecture for unsupervised parsing

A cascade of chunkers is a sequence wherein the output of one provides input to the next. Let $M^{(i)}$ denote a chunking model in such a sequence; so $M^{(0)}$ is simply an unsupervised chunker, taking as input raw text – possibly different texts for training and evaluation (or run-time) – and producing bracketings over the raw text used as evaluation. $M^{(1)}$ describes the next chunker in the cascade, or the model at *level 1*, *etc.*

Training data for the $M^{(1)}$ chunker is provided by the previous model in the cascade. When $M^{(0)}$ is trained, it is then run on the training data provided (call it $\mathbf{x}^{(0)}$), chunking the training data. The chunked training data is then converted into a new dataset to train $M^{(1)}$ as follows: the chunks identified are converted into new symbols I call *phrasal stand-ins*. Let $\mathbf{x}^{(1)}$ denote this new dataset; $\mathbf{x}^{(1)}$ now resembles the original raw text training corpus, in that it is a sequence of symbols, only now mixed in to the sequence of raw text words are new symbols in the places where $M^{(0)}$ identified chunks. This new dataset is used as training material for $M^{(1)}$, and this process repeats: each model $M^{(i)}$ takes as training input the converted output of the previous model $\mathbf{x}^{(i)}$, and chunks it and creates a new training set $\mathbf{x}^{(i+1)}$ which is used to train a new chunking model $M^{(i+1)}$. This process is repeated until there is no change in model output, that is until $\mathbf{x}^{(i)} = \mathbf{x}^{(i+1)}$.

Selecting phrasal stand-ins A crucial step in this process is the selection of new symbols to stand in for the phrases identified by a chunker $M^{(i)}$ in the cascade. The strategy adopted for this is quite simple: say, in the original training set $\mathbf{x}^{(0)}$, "the brown bear" is identified as a chunk. To convert this into a phrasal stand-in, a word is selected from those of the chunk to represent the phrase, and marked to indicate this is a phrasal stand-in. The heuristic to select a word is simple: choose the word with the highest frequency in $\mathbf{x}^{(0)}$. In the example, this word would (almost

$$
\begin{array}{lccccccc}
\mathbf{x}^{(0)} = & \ldots & \text{chairman} & \text{of} & \text{the} & \text{board} & \ldots \\
\mathbf{x}^{(1)} = & \ldots & \text{chairman} & \text{of} & \boxed{\text{the}} & & \ldots \\
\mathbf{x}^{(2)} = & \ldots & \text{chairman} & \boxed{\boxed{\text{the}}} & & & \ldots
\end{array}
$$

Figure 4.1: Example term updates in cascaded chunker training data.

certainly) be *the*, so phrasal stand-in for this term is[1]

the

I also call these *pseudowords*, since they are treated as words in the training of the next chunker model, but are not exactly words from the original corpus.

At any level i, the phrasal stand-ins for $\mathbf{x}^{(i+1)}$ are generated using the corpus frequencies from $\mathbf{x}^{(i)}$. Thus, pseudowords can themselves can be used to represent phrases. Consider, for example Figure 4.1. In this sense, the only interaction between the models is the data passed from one level to the next.

This is, of course, a very simple method for selecting phrasal stand-ins, but for the purposes of cascaded parsing it works well. Very likely, a more sophisticated method for this step may well lead to improvements in performance.

Linguistic note This heuristic – choose the term with the highest corpus frequency – is only really meant to enable the parsing process outlined here, and not have broader linguistic implications. Having said that, it may be construed as a kind of simple or baseline method for selecting the head of a phrase. As such, in the case of noun phrases containing a determiner, it in effect subscribes to the tradition that postulates the head of a noun phrase is the determiner. This has been the tradition of categorial and type-logical grammars (Moortgat, 1997; Steedman, 1996, 2001), opposed by earlier traditions of generative grammar (Chomsky,

[1] I use boxes to convey marking this symbol as a phrasal stand-in for the phrase; in the program managing this process, phrasal stand-ins are marked simply by appending = to the text of the word chosen to represent the phrase. So, given following string is chunked as follows

```
(on sunday) , (the brown bear) sleeps
```

this is converted to

```
=on , =the sleeps
```

I mention this simply to point out that the computational method implemented and used to evaluate these ideas really is as simple as described in this chapter.

1965) wherein the head of an NP is taken to be the noun. (In more recent work in the generative grammar, for example Abney, 1987 and Radford, 2004, this has switched: determiners are basically heads of NPs.)

On the other hand, this strategy had the effect, in English, of distinguishing all of the NPs headed with *the* from those headed with *a* (and, separately still, those with *an*). This is not wholly unreasonable: indefinite noun phrases (those with *a*/*an*) serve a different discourse function than definite noun phrases (with *the*): in the simplest case, indefinite noun phrases introduce a new entity into the universe of discourse, whereas definite noun phrases refer back to previously introduced entities (Heim, 1982; Kamp and Reyle, 1993). There are, of course, a wide range of related issues and several generations of linguistics research into this aspect of discourse semantics. For the present purposes, though, the net effect is that definite noun phrases tend to occur at the beginnings of sentences, and indefinite noun phrases after the beginning.

Unfortunately, this heuristic does not provide as clean a linguistic generalization at subsequent levels of cascaded chunking: Figure 4.1 is a realistic example, even though it is made up, in that in the English cascaded chunking experiments reported below, in the second cascade dataset ($\mathbf{x}^{(1)}$) the frequency of the pseudoword $\boxed{\textit{the}}$ (the phrasal stand-in for chunks containing *the*) has higher frequency than the term *of*.

Parsing For parsing, the cascade of chunkers $(M^{(0}, M^{(1)}, \ldots)$ builds hierarchical constituents bottom-up. Similar to training, the raw text to be parsed is initially chunked by $M^{(0)}$, then the chunked sentence is converted to a new sequence by replacing predicted chunks with pseudowords, using the same strategy and corpus frequencies from the original training material, $\mathbf{x}^{(0)}$. The converted sequence is then chunked by $M^{(1)}$, and this process is repeated. During the parsing process, the process remembers original chunks predicted at each level, so when complete it is simple to unwind the phrasal stand-ins to produce a full constituent tree. This is illustrated in Figure 4.2.

This parsing strategy does not guarantee that a root bracket spanning the full sentence will be predicted by one of the chunkers; a root bracket is added to the tree output at the end of the parsing process. However, looking at parser output, in the absense of phrasal puncutation a bracket spanning the full sentence usually

1. Start with raw text:

there is no asbestos in our products now

2. Apply chunking model:

there (*is no asbestos*) *in* (*our products*) *now*

3. Create pseudowords:

there $\boxed{\textit{is}}$ *in* $\boxed{\textit{our}}$ *now*

4. Apply chunking model (and repeat 3–4):

(*there* $\boxed{\textit{is}}$) (*in* $\boxed{\textit{our}}$) *now*

5. Unwind and create a tree:

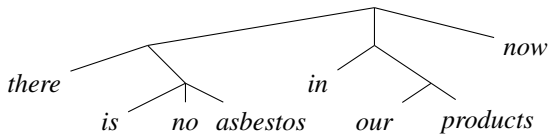


Figure 4.2: Cascaded chunking illustrated. Pseudowords are indicated with boxes.

is predicted by one of the chunkers in the cascade.

4.3 Evaluation.

Each chunker in the cascade chunks the raw text, then regenerates the dataset replacing chunks with pseudowords; this process is iterated until no new chunks are found. This process is applied to the training dataset independently of its application to the evaluation dataset. The same training, development and held-out test datasets described in §2.8 are used here. The separate chunkers in the cascade are referred to as *levels* – in our experiments the cascade process took a minimum of 5 levels, and a maximum of 7. All chunkers in the cascade have the same settings in terms of smoothing, the tagset and initialization. Phrasal punctuation is kept in each level of the cascade, and only removed before the final trees are created – this

	WSJ			Negra			CTB		
	Prec.	Rec.	*F*	Prec.	Rec.	*F*	Prec.	Rec.	*F*
Random	*24.0*	*30.9*	*27.0*	*20.1*	*38.8*	*26.5*	*28.2*	*32.7*	*30.3*
Baseline	*44.5*	*57.4*	*50.1*	*19.5*	*37.7*	*25.7*	*35.6*	*41.2*	*38.2*
CCL	53.6	**50.0**	51.7	33.4	32.6	33.0	37.0	21.6	27.3
HMM*	48.2	43.6	45.8	30.8	**50.3**	38.2	43.0	29.8	35.2
PRLG*	**60.0**	49.4	**54.2**	**38.8**	47.4	**42.7**	**50.4**	**32.8**	**39.8**

Table 4.1: Parsing evaluation on test sets: unlabeled PARSEVAL of baselines, CCL and cascaded chunkers. The right-branch baseline uses phrasal punctuation as described in Chapter 2 (see Figure 2.11 on p. 37).

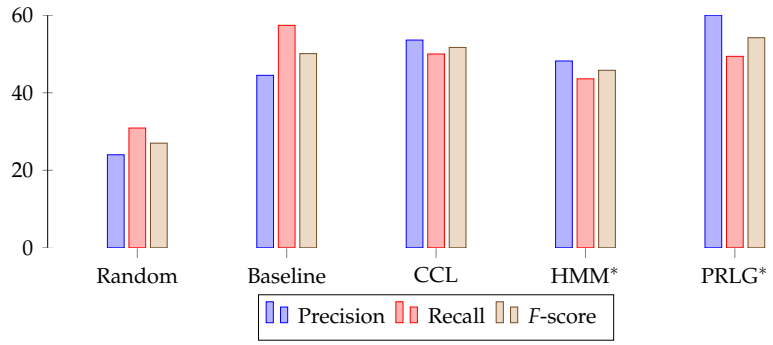
is analogous to the default behavior of CCL parser.

4.3.1 Basic results

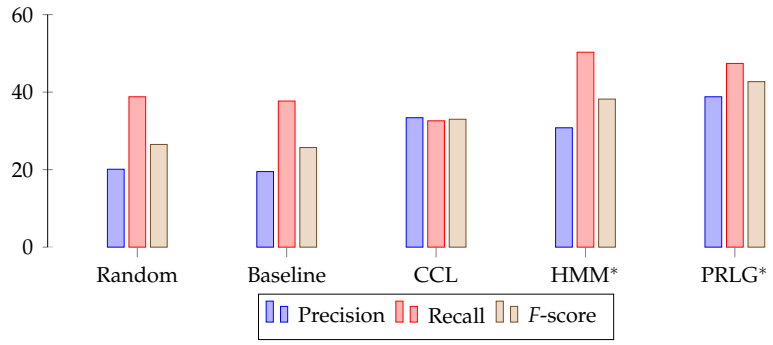
Table 4.1 gives the unlabeled PARSEVAL scores for CCL and the two basic finite-state models from §3.3.1. These are the HMM and PRLG taggers, using the basic BIO tagset. PRLG achieves the highest *F*-score for all datasets, and does so by a wide margin for German and Chinese. CCL does achieve higher recall for English.

Baselines The random and right-branching baselines from Chapter 2.7.2 are also given in Table 4.1. For the right-branching baseline, I use the phrasal punctuation-sensitive right-branching baseline, which does not branch over phrasal punctuation (*c.f.* §2.7.1), which is stronger than the simple right-branching baseline. The random baseline consists of randomly constructed binary trees, the numbers quoted here are the average of 100 runs of this baseline experiment. The results of these baselines, the CCL benchmark, and the sequence models are plotted in Figure 4.3.

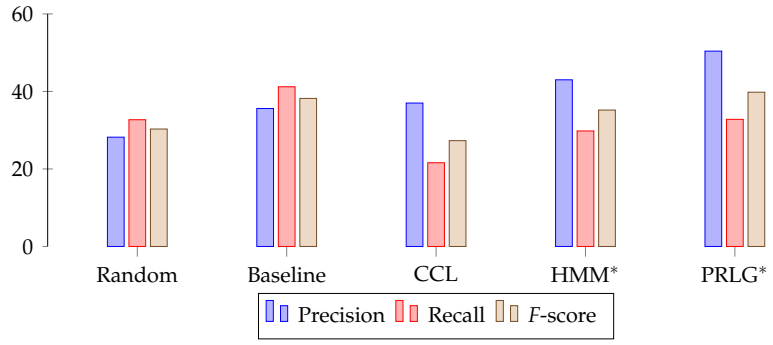
NPs and PPs recall While the first level of constituent analysis has high precision and recall on NPs, the second level often does well finding prepositional phrases (PPs), especially in WSJ; see Table 4.2. This is illustrated in Figure 4.4. This example also illustrates a PP attachment error, which are a common problem for these models.



a) WSJ



b) Negra



c) CTB

Figure 4.3: Bar charts for full sentence parsing performance: baselines, benchmark and cascaded sequence models.

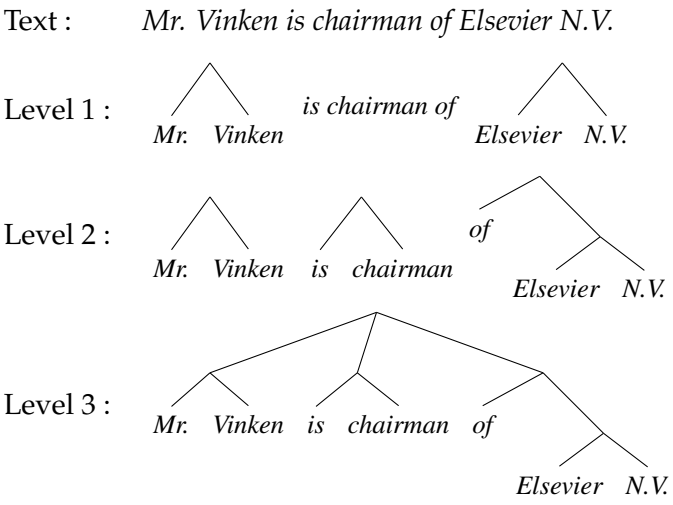


Figure 4.4: PRLG cascaded chunker output.

Cascaded parsing trade-offs There is a trade-off involved with building constituent trees in a cascaded fashion, as here. The lower levels, by design, have relatively high precision at constituent identification, however as more constituents are added, the precision of these newly predicted constituents goes down as the overall recall of the parser goes up. In part, this is due to the fact that the sequence models perform better on chunking actual strings of terms, as in basic UPP, than chunking strings with a mixture of natural language terms and pseudowords standing in for phrases. The former seems to be a more natural task for unsupervised chunking.

The performance of the different models at each level in of the cascaded chunking process is illustrated in figures 4.5 to 4.7. The graphs for WSJ typify the precision/recall trade-off. The plots for Negra and CTB are more surprising, in certain respects.

For Negra, while, as expected, the precision of the first level chunker (that is, the unsupervised chunker from Chapter 3) is much higher than recall, this relationship switches from level 2 on. This pattern holds for both models. Very likely, this is due in large part to the overall far fewer constituents in Negra versus the others: see Table 4.3 (repeating in part Table 2.1 from p. 24). So, the cascaded sequence models, which tend to predict a roughly constant number of constituents per word (roughly .20–.25 constituents per word) will predict relatively more constituents relative to the gold standard than in the other datasets.

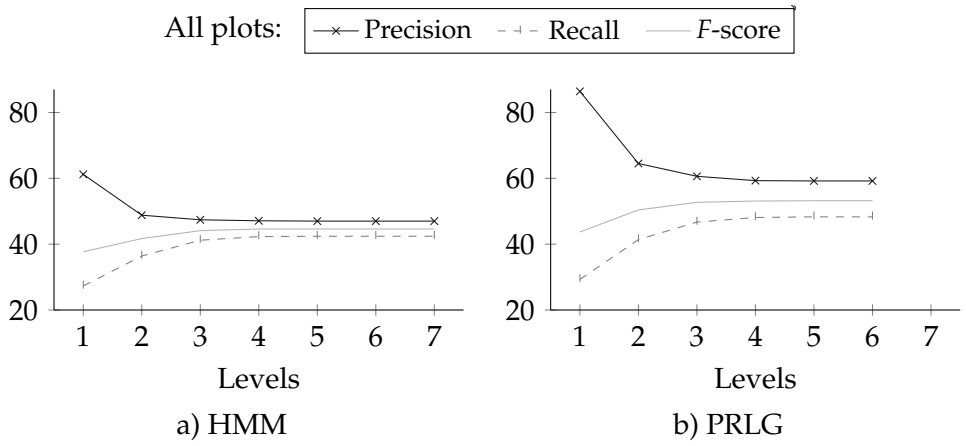


Figure 4.5: Precision and recall of cascaded parsers at successive levels: **WSJ**.

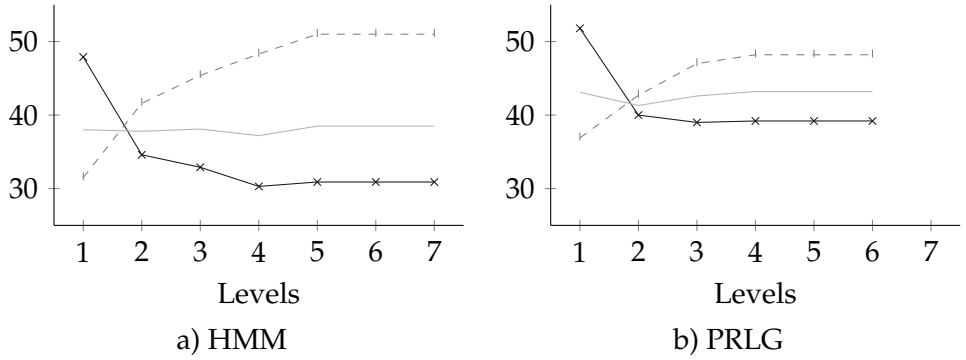


Figure 4.6: Precision and recall of cascaded parsers at successive levels: **Negra**.

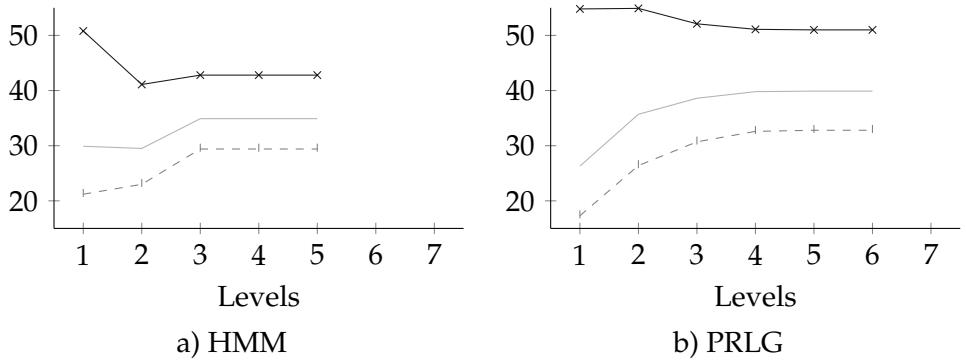


Figure 4.7: Precision and recall of cascaded parsers at successive levels: **CTB**.

		NPs		PPs	
		Lev 1	Lev 2	Lev 1	Lev 2
WSJ	HMM	66.5	68.1	20.6	70.2
	PRLG	77.5	78.3	9.1	77.6
Negra	HMM	54.7	62.3	24.8	48.1
	PRLG	61.6	65.2	40.3	44.0
CTB	HMM	33.3	35.4	34.6	38.4
	PRLG	30.9	33.6	31.6	47.1

Table 4.2: NP and PP recall at cascade levels 1 and 2. The level 1 NP numbers differ from the NP chunking numbers from Table 3.3 since they include root-level constituents which are often NPs.

		WSJ	Negra	CTB
constituents per sentence	Gold standard	14.5	6.8	17.2
	HMM	5.5	3.3	6.2
	PRLG	4.0	3.6	4.4
constituents per sent. normalized	Gold standard	14.2	9.4	15.4
	HMM	5.4	4.6	5.4
	PRLG	4.0	5.0	4.0

Table 4.3: Constituent statistics for each corpus: the gold standard annotations and output of the cascaded chunking models. The normalized average constituents per sentence is the average number of constituents per word x 20; this is to adjust for the fact that Negra sentences (avg. 15.1 words) are shorter than WSJ (avg. 21.4) and CTB (avg. 21.7).

The sequence models' performance at the different cascade levels on CTB shows interesting patterns as well. The cascaded HMM chunker model precision actually improves after level 2; likewise, the PRLG chunker model precision never degrades very much. This pattern is addressed in §4.3.4: it turns out that, while phrasal punctuation provides limited help to the sequence models at the constituent chunking task – *c.f.* Table 3.14a – it provides a significant advantage to cascading parsing strategies, especially for CTB. In other words, for CTB, fairly soon in the cascade of chunkers the model has identified brackets as delimited by phrasal punctuation; and, this turns out to be a relatively good heuristic, as these brackets have high precision.

4.3.2 Evaluation on short sentences

The proposed models are also evaluated using short – 10-word or less – sentences. That said, the training/test split from before remains. Also, making use of the open source implementation by F. Luque,[2] we compare on WSJ and Negra to the *constituent context model* (CCM) of (Klein and Manning, 2002, also, §2.6.1). CCM learns to predict a set of brackets over a string (in practice, a string of POS tags) by jointly estimating constituent and distituent strings and contexts using EM. Some points about this comparison bear raising: on the one hand, CCM is evaluated using gold standard POS sequences as input, so it receives a major source of supervision not available to the other models. On the other hand, the other models use punctuation as an indicator of constituent boundaries, but all punctuation is dropped from the input to CCM. Also, note that CCM performs better when trained on short sentences, so here CCM is trained only on the 10-word-or-less subsets of the training datasets.[3] CCM training is run for 50 iterations of EM. Results are reported in Table 4.4.

The results from the cascaded PRLG chunker are near or better than the best performance by CCL or CCM in these experiments. These and the full-length parsing results suggest that the cascaded chunker strategy generalizes better to longer sentences than does CCL. CCM does very poorly on longer sentences, but does not have the benefit of using punctuation, as do the raw text models. Also reported is a variation of CCM, constrained to recognize phrasal punctuation and use it in a manner similar to raw text models. This has a positive impact on CTB results, but not the other data-sets. Finally, note that CCM has higher recall, and lower precision, generally, than the raw text models. This is due to the fact that CCM is defined only for binary branching trees. CCL and the cascaded models can predict higher-branching constituent structures, so fewer constituents are predicted overall.

[2] Luque's CCM code is available at

`http://www.cs.famaf.unc.edu.ar/~francolq/en/proyectos/dmvccm/`.

The results in this chapter are based on a fork of this project to support the corpora evaluated here, as well as phrasal punctuation. This is hosted at

`https://bitbucket.org/eponvert/pyccm`.

Both versions are released under the GNU General Public License.

[3] This setup is the same as Seginer's (2007a), except the train/test split.

	WSJ			Negra			CTB		
	Prec.	Rec.	*F*	Prec.	Rec.	*F*	Prec.	Rec.	*F*
Baseline	*60.3*	*77.7*	*67.9*	*92.0*	*42.0*	*57.7*	*49.3*	*66.8*	*56.7*
CCM	*62.4*	*81.3*	*70.6*	*51.2*	*87.4*	*64.5*	*39.9*	*54.1*	*45.9*
CCM^P	*60.4*	*71.4*	*65.4*	*51.4*	*84.0*	*63.8*	*49.0*	*61.7*	*54.6*
CCL	71.2	**73.1**	**72.1**	52.9	54.0	53.0	54.4	44.3	48.8
HMM*	64.4	64.7	64.6	47.7	72.0	57.4	55.8	53.1	54.4
PRLG*	**74.6**	66.7	70.5	**56.3**	**72.1**	**63.2**	**62.7**	**56.9**	**59.6**

Table 4.4: Evaluation on 10-word-or-less sentences. CCM scores are italicized as a reminder that *CCM uses gold-standard POS sequences as input*, so its results are not strictly comparable to the others. CCM^P refers to the CCM parser using phrasal punctuation similarly to CCL and the sequence models. HMM* and PRLG* are the cascaded sequence models.

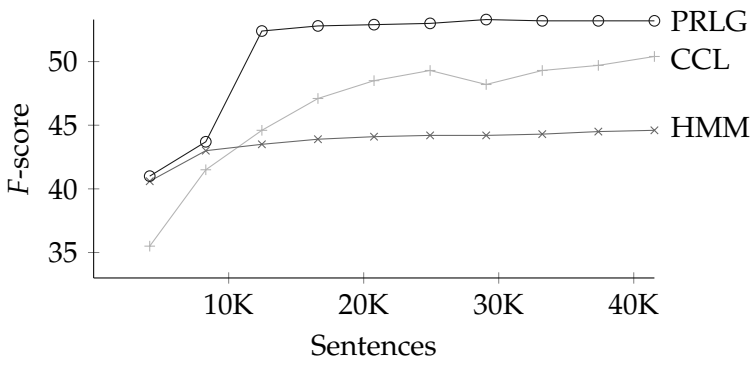
4.3.3 Learning curves

The learning curves for the cascaded chunking models are given, with learning curves for the CCL benchmark, in Figure 4.8.

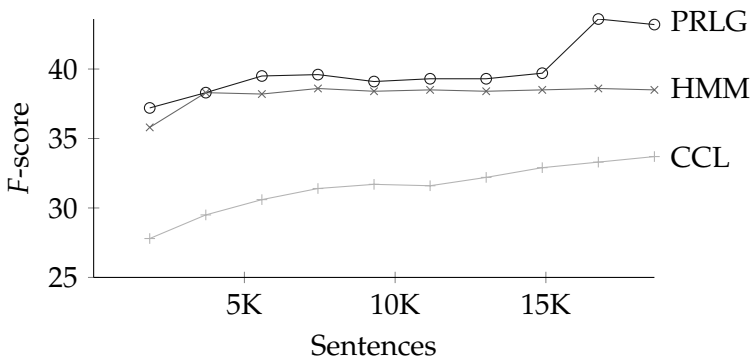
For WSJ, the cascaded PRLG model fairly quickly converges to close to its final performance. I identify two reasons for this: firstly, in comparison to the PRLG constituent chunking learning curve from Figure 3.7 (p. 66), both models achieve close to their peak performance at roughly the same point in the training data. Secondly, due to the cascaded chunking strategy, much of the long-tail of vocabulary items is consumed by the process of creating pseudowords, so at levels beyond the initial chunking, larger amounts of training data do not result in very different higher level constituent predictions.

This basically characterizes the cascaded PRLG for Negra parsing, as well. The jump in cascaded PRLG performance corresponds to a similar jump in the PRLG chunker in that region of the training material. For CTB, the best point of comparison is Figure 3.20c (p. 93) which suggests a smooth, gradual improvement in model performance, as the cascaded PRLG is in this plot.

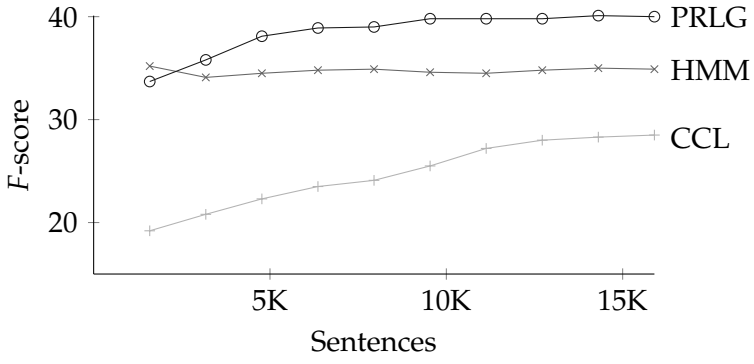
In all three datasets, the CCL model seems to be improving more gradually but steadily than the cascaded parsers are. This is possibly due to the nature of the models being learned. Whereas the cascaded models are learning parameters to sequence models, with these they learn fairly general, but effective, patterns in the



a) WSJ



b) Negra



c) CTB

Figure 4.8: Learning curves for CCL and cascaded parsing models.

		WSJ			Negra			CTB		
		Prec.	Rec.	*F*	Prec.	Rec.	*F*	Prec.	Rec.	*F*
CCL	W/ Punctuation	53.6	50.0	51.7	33.4	32.6	33.0	37.0	21.6	27.3
	No Punctuation	39.5	36.5	37.9	22.7	28.0	25.1	25.4	15.6	19.3
	Difference	−14.1	−13.5	−13.8	−10.7	−4.6	−7.9	−11.6	−6.0	−8.0
HMM	W/ Punctuation	48.2	43.6	45.8	30.8	50.3	38.2	43.0	29.8	35.2
	No Punctuation	40.4	35.0	37.5	28.0	52.0	36.4	29.6	28.6	29.1
	Difference	−7.8	−8.6	−8.3	−2.8	+1.7	−1.8	−13.4	−1.2	−6.1
PRLG	W/ Punctuation	60.0	49.4	54.2	38.8	47.4	42.7	50.4	32.8	39.8
	No Punctuation	49.9	42.2	45.7	34.8	42.9	38.4	28.4	21.0	24.1
	Difference	−10.1	−7.2	−8.5	−4.0	−4.5	−4.3	−22.0	−11.8	−15.7

Table 4.5: Evaluation of parsing models with and without phrasal punctuation. Without phrasal punctuation, only sentence boundaries are treated as boundary symbols (*i.e.* have tag STOP).

data, CCL is learning a lexicon, and with it a separate set of statistics for each word. It may well be possible that CCL will benefit from greater exposure to infrequently seen vocabulary and textual data.

4.3.4 Phrasal punctuation revisited

Up to this point, the cascaded parsing models use phrasal punctuation as a phrasal separator, like CCL. Table 4.5 compares these results to the parser run without phrasal punctuation in training and evaluation.

The results for cascaded parsing differ strongly from those for chunking. Using phrasal punctuation to constrain bracket prediction has a larger impact on cascaded parsing results almost across the board. This is not surprising: while performing unsupervised partial parsing from raw text, the sequence models learn two general patterns: i) they learn to chunk rare sequences, such as named entities, and ii) they learn to chunk high-frequency function words next to lower frequency content words, which often correlate with NPs headed by determiners, PPs headed by prepositions and VPs headed by auxiliaries. When these patterns are themselves replaced with pseudowords (see Figure 4.2), the models have fewer natural cues to identify constituents. However, within the degrees of freedom allowed by punctuation constraints as described, the chunking models continue to find relatively

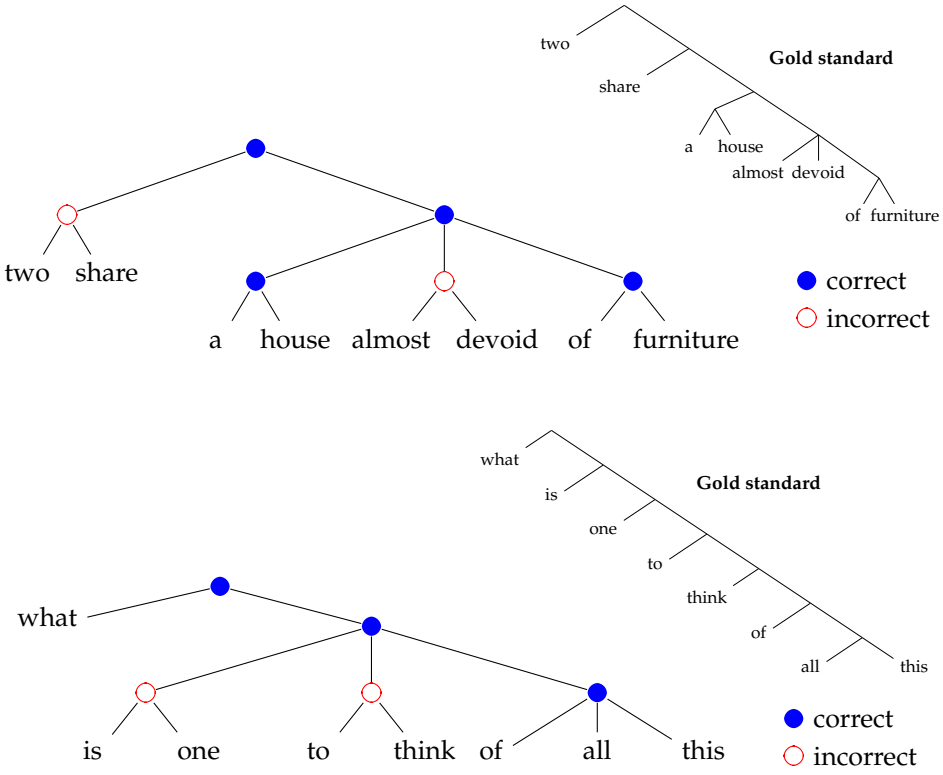


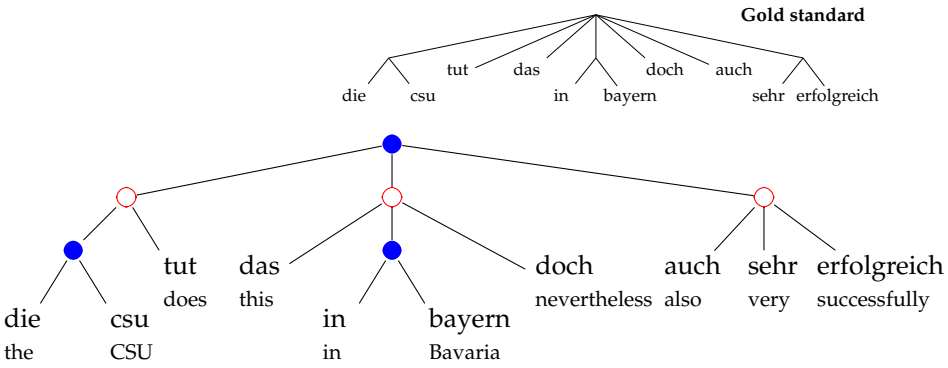
Figure 4.9: Example output of the cascaded PRLG chunker: **WSJ**.

good constituents.

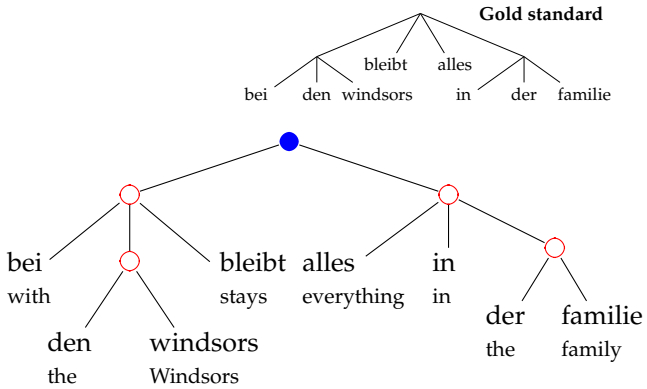
While CCL makes use of phrasal punctuation in previously reported results, the open source implementation allows it to be evaluated without this constraint. Table 4.5 reports results of CCL run on the same data, without phrasal punctuation in training or evaluation material. CCL is, it turns out, very sensitive to phrasal punctuation. Comparing CCL to the cascaded chunkers when none of them use punctuation constraints, both HMMs and PRLGs outperform CCL for each evaluation and dataset.

4.3.5 Example output

Example parser output is given for WSJ in Figure 4.9 and for Negra in Figure 4.10 and Figure 4.11. With these, the fact that the cascaded parsing models produce far fewer constituents than are present in the gold standard is visually apparent. Also,



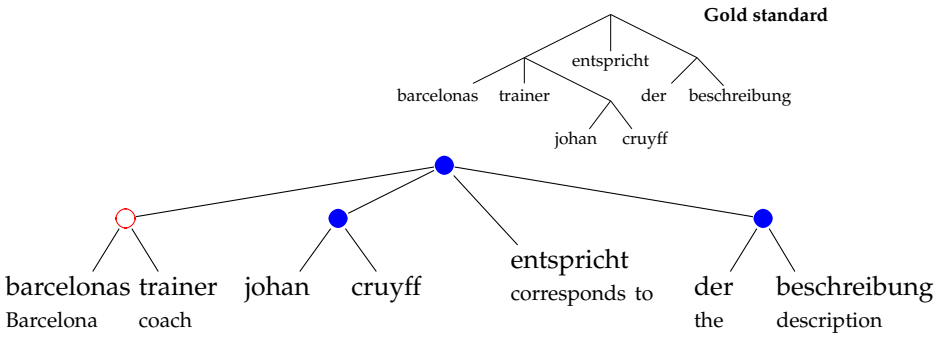
Nevertheless, the CSU does this in Bavaria very successfully as well



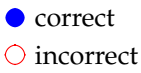
With the Windsors everything stays in the family.

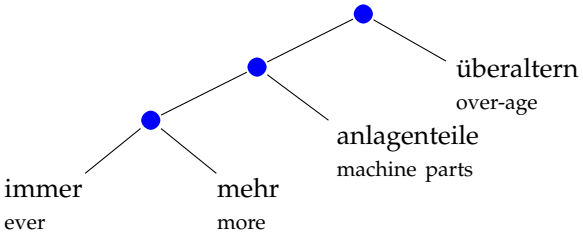
correct
incorrect

Figure 4.10: Example output of the cascaded PRLG chunker: **Negra**.



Barcelona coach Johan Cruyff corresponds to the description





(with) more and more machine parts over-age

Figure 4.11: More example output of the cascaded PRLG chunker for Negra.

the tendency of the models to chunk the subject and the verb – especially when the subject is short, like a pronoun – is also indicated in this example output. These samples also make clear the consequences of favoring local constituent prediction over global models: in the second example in Figure 4.9, the local prediction of the constituent (*to think*) makes it impossible for the model to get the branching structure right in (*to think of all this*); similarly for the local constituent prediction (*is one*).

In the Negra examples, similar patterns emerge. However, the negative results stemming from the unannotated noun phrases in the Negra gold standard also emerge: in the second example in Figure 4.10, the NPs *den windsors* and *der familie* are not annotated as such, leading to false positives for these constituents. On the other hand, Figure 4.11 provides some examples where the structures predicted by the cascaded models intersect better with the gold standard.

4.3.6 Computational analysis

Since at the core of the cascaded parsing strategy is a set of unsupervised partial parsing models, the computational analysis of those models, §3.3.9, applies to these as well.

In addition, the number of overall levels required to train a cascaded model is arbitrary: this is $\log K$ for K the length of the longest sentence; more specifically, the base of the logarithm is related to the branching factor of the tree model, though this is not constant across datasets or trees. In practice, 5 to 7 levels total are required for the datasets explored in this work, *c.f.* Figure 4.5 to Figure 4.7. Because at each level in the cascade, multi-word chunks are replaced with single pseudowords, the length N of the dataset to be tagged by the next sequence model decreases, approaching S the number of sentences.

4.4 Historical connections

This kind of parsing architecture – cascaded constituent prediction models – has been implemented in one form or another at different times in the history of computational linguistics. I highlight a few instances here.

The use of cascaded finite-state models for hierarchical structural predic-

tions is – interestingly – one of the oldest strategies in computational linguistics/natural language processing, dating to the work of Zellig Harris on the Transformations and Discourse Analysis Project (TDAP) in late 1950s and 1960s. Joshi and Hopely (1996) describe reimplementing the TDAP parser, as follows:

> The program was essentially a cascade of Finite state transducers (FSTs).[4] To the best of our knowledge, this is the first application of FSTs to parsing. The program consisted of the following phases:
>
> 1. Dictionary look-up.
> 2. Replacement of some 'grammatical idioms' by a single part of speech.
> 3. Rule based part of speech disambiguation.
> 4. A right to left FST composed with a left to right FST for computing 'simple noun phrases'.
> 5. A left to right FST for computing 'simple adjuncts' such as prepositional phrases and adverbial phrases.
> 6. A left to right FST for computing simple verb clusters.
> 7. A left to right 'FST' for computing clauses.

Technically, the TDAP system was supervised, since it required hand-engineered rules, but it could not take advantage of treebank annotations since, in the late 1950s, there were no electronic treebanks. Interestingly, the cascaded models in this work implicitly mirror much of Harris and his colleagues' strategy: identification of simple noun phrases, identification of prepositional phrases, and the cascaded pipeline process in general.

In more recent work, Brooks (2006) reports an unsupervised parsing strategy that is similar to that proposed here. Specifically, he uses a set of frequency based heuristics to choose local constituents – chunkers, basically – then uses a cascaded strategy to produce constituent tree structures, like this work. The major difference between that work and this is the chunking strategy employed: while in the current work, sequence models are used a tagset encoding local constituent structure, Brooks uses a per-word heuristic to decide if a word was likely to begin or end

[4] efp: Notably, finite-state transducers are closely theoretically connected to hidden Markov models, see Jurafsky and Martin (2008).

a phrase, by generalizing on whether or not the word was likely to begin or end a sentence. Unfortunately, Brooks' empirical evaluation suggests this heuristics did not manage to produce a parsing strategy competitive with a right-branching baseline. On the other hand, Brooks did not make use of phrasal punctuation, as did Seginer (2007a), which turns out to be a very helpful source of phrasal boundary information.

Brants (1999) reports a supervised cascaded chunking strategy for parsing which is also strikingly similar to the methods proposed here. In both, Markov models are used in a cascade to predict hierarchical constituent structure; and in both, the parameters for the model at each level are estimated independently. There are major differences, though: the models here are learned from raw text without tree annotations, using EM to train parameters; Brants' cascaded Markov models use supervised maximum likelihood estimation. Secondly, between the separate levels of the cascade, in this work constituents are converted into symbols which are treated as tokens in subsequent chunking levels; the Markov models in the higher cascade levels in Brants' work actually emit constituent structure. A related approach is that of Schuler et al. (2010), who report a supervised hierarchical hidden Markov model which uses a right-corner transform. This allows the model to predict more complicated trees with fewer levels than in Brants' work or this paper.

4.5 Summary

This chapter showed that the unsupervised partial parsing models can be applied in a cascaded fashion to produce full unlabeled constituent parse trees from the bottom up. The basic strategy is to convert the output of a partial parser – a chunked corpus – and replace the chunks with pseudoword symbols chosen to represent the chunk, then chunk the resulting corpus and repeat.

This strategy outperforms the CCL unsupervised parser in most evaluations, and for short sentences is even competitive with the CCM unsupervised parser, even though the latter is trained and evaluated using gold standard POS.

Reconsidering the use of phrasal punctuation in the parsing setting, it turns out that its influence on this parsing strategy is stronger than it is on text chunking. However, it has a stronger effect still on the performance of the CCL parser of Seginer (2007a), who introduced this usage of punctuation.

Chapter 5

Conclusion

This dissertation has presented an approach to unsupervised constituent parsing which is simple, efficient, easily understood, yet nevertheless achieves best or near-best results on commonly used datasets in English, German and Chinese. The idea is to focus on the prediction of local constituents with high-precision.

In Chapter 3 this task is presented in isolation, called unsupervised partial parsing, the unsupervised analog of text chunking. Two evaluations of unsupervised partial parsing are proposed in this work, motivated and discussed: constituent chunking and base NP identification. Both are based on gold standard treebank annotations, and so complement treebank-based parser evaluation. Also, especially the unsupervised base NP identification task is well-motivated on its own, as it is connected (at least thematically) to supervised NP identification research (Abney, 1991; Ramshaw and Marcus, 1995; Tjong Kim Sang and Buchholz, 2000), and has the potential to contribute to applications in query segmentation, text classification, coreference and semantic role labeling (see §2.3.3 on p. 17).

Many of the most successful approaches to chunking have made use of sequence models predicting tags which encode non-overlapping constituents. A familiar one is BIO tagging, where a word tagged B is taken to be the beginning of a constituent, a word tagged I is taken to be inside (but not beginning) a constituent, and a word tagged O is understood to not be in a constituent. Learning tagging models has a long and successful history in computational linguistics research, and there are both supervised and unsupervised approaches. One of the best known unsupervised approaches is learning *hidden Markov models* with expectation max-

imization. These models are applied to unsupervised chunking, and work quite well, beating the benchmark for German and Chinese evaluation. Another model class explored in this work, a variation of hidden Markov models called probabilistic right linear grammars, works better still for evaluation on English – for chunking, this model outperforms strong baseline based on gold-standard part of speech tags, as well as a supervised HMM tagging model benchmark.

Models for unsupervised partial parsing may be used to constrain other (third-party) approaches to unsupervised parsing, or other applications for natural language processing. However, these models may also be used in a relatively direct fashion to produce full parse trees, as presented in Chapter 4. The parsing strategy employed uses a *cascade of chunkers*, where each chunker (other than the first) takes as input the output of the previous, with the chunks predicted by the previous chunker encoded as *phrasal stand-ins* or *pseudowords*. Keeping it simple, the method adopted is to choose the term within the chunk with the highest corpus frequency to represent the chunk.

When a PRLG model is the chunking technique, this parsing strategy outperforms a current state-of-the-art for raw text unsupervised parsing (CCL Seginer, 2007a), for all three datasets. When an HMM is the underlying chunking model, this parsing strategy still beats the CCL benchmark. These models are also competitive when evaluated on short sentences, and are competitive with, if not superior to the CCM model of Klein and Manning (2002), even though the latter has access to gold-standard part-of-speech symbols (POS) at training and evaluation.

5.1 Progress toward our goals

Returning to the goals for unsupervised parsing sketched in §1.2, let us review the progress made.

1. Operate on raw text The sequence models for unsupervised partial parsing, and the cascaded models for parsing, are developed for and evaluated on raw text corpora. These models' performance for chunking and parsing is, at its best, as good or better than models or strategies which can take advantage of gold standard POS. For base NP identification in English, the PRLG model matches the GPOS-$\star$ benchmark, which chunks noun phrases based

on POS sequences known to correspond with NPs in the training data; for parsing Chinese and German, the cascaded models outperform the CCM parser, which learns a constituency model over strings of POS. These results do not generalize to all the tasks sketched in this work, nevertheless some progress has been made toward raw-text constituent structure induction.

2. **Extensible** Despite its several positive points, a frustrating aspect of Seginer (2007a)'s CCL parser is that its parsing strategy is complicated set of interconnected heuristics, making it difficult to re-implement, extend, or try to build on with alternative heuristics or statistical techniques. The strategy proposed in this thesis, on the other hand, uses standard and well-understood techniques from natural language processing: text chunking, as an approach to partial parsing, dates back at least to Abney (1991); the treatment of chunking as a tagging problem, using a BIO-style tagset, dates to Ramshaw and Marcus (1995); the use of hidden Markov models for many applications in speech and language processing has become ubiquitous in the field (Jurafsky and Martin, 2008); and the use of cascaded sequence models for hierarchical structural predictions is – interestingly – one of the oldest strategies in computational linguistics/natural language processing, dating to the work of Zellig Harris on the Transformations and Discourse Analysis Project in late 1950s and 1960s (Joshi, 2002). All of this is to say, the component parts of the strategies proposed in this work have a track record in computational linguistics, and many have extensive foundational research. For instance, in addition to HMMs, just one other sequence model class for tagging is used in this work: the PRLG. While this model achieves many of our best results, many others remain untried. For instance, one could employ (unsupervised) discriminative learning techniques with features (Berg-Kirkpatrick et al., 2010), integer programming (Ravi and Knight, 2009), or incorporate topic models and document information (Griffiths et al., 2005; Moon et al., 2010), just to name a few. Moreover, **upparse**, the software used for the experiments in this research, is available under an open-source license to facilitate replication and extensions.[1]

3. **Efficient** Computational analyses are provided in earlier chapters; basically, be-

[1] `http://elias.ponvert.net/upparse`.

cause the core learning process makes use of the Baum-Welch algorithm, learning time and decoding (chunking) is linear with data-set length. The cascaded strategy adds another factor, in that the chunker-learner strategy must be iterated once for each level in the cascade; this factor is the log of the length of the longest sentence in the dataset – 5 to 7 in experiments in this work.

Seginer's CCL parser also scales linearly with dataset length. The parser is incremental and linear-time with respect to sentence length. In practice, to produce full trees, CCL ran faster than the cascaded models; for just chunking, *i.e.* a single level in the cascade, the sequence models are more competitive – though of course CCL is learning a parsing model, not just chunking. For example, on consumer grade hardware,[2] a cascaded parsing experiment for WSJ, using PRLG models – learning from the approximately 40K sentences and 800K words – took a little over 13 minutes (13:26), whereas the CCL parser took about 4:30 (again, minutes) on the same datasets; for just chunking, the PRLG converges at 78 iterations of EM at about 7:10.

This is in contrast to many parsing and parser learning methods, which operate in polynomial time with respect to sentence length. CCM is one such method: its EM method, which is conceptually similar to the inside-outside method, operates in $O(n^3)$ for length of sentence n (Klein, 2005). Anecdotally, the implementation used for experiments in this dissertation (*c.f.* Chapter 4) runs quickly for datasets with short sentences, but requires hours for datasets with up to just 30 word sentences. Moreover, CCM performance in comparison to treebank annotations degrades for sentence lengths >10.

In short, the methods proposed for unsupervised partial parsing in this work are sufficiently efficient in time and space to be useful for practical purposes. Moreover, there is a standard technique to implement hidden Markov model training in a potentially massively parallel fashion, using the Map-reduce computing paradigm (Lin and Dyer, 2010). This technique applies straightforwardly to PRLGs as well.

4. State of the art performance Finally, at least for the experiments reported in this

[2] An Apple MacBook Air with a 1.6GHz Intel Core 2 Duo processor; the unsupervised partial parsing code requiring less than 1G RAM.

work, unsupervised partial parsing and cascaded chunker parsing nearly match or, usually, exceed current state-of-the-art approaches. These experiments use standard datasets and evaluations, which have been used by Seginer, Klein and Manning and others in comparable analyses.

The techniques and results reported in this dissertation makes solid progress toward our goals for unsupervised parsing research.

5.2 Remaining issues

In spite of the progress made in this work, several issues remain.

Phrasal punctuation Phrasal punctuation has special treatment in the methods in this work – the same treatment, basically, they are given by the CCL parser. This involves altering the raw text of the corpus, and some language data – such as spoken language data – will not even have punctuation. While simply dropping phrasal punctuation has a stronger negative effect on CCL than on the cascaded chunking strategy, and while (first-level) chunking models seem less affected by ignoring punctuation than cascaded chunkers, all the same it seems clear that this heuristic has a strong positive impact on parsing and chunking results. In future work, if we could alter these models to operate without reliance on this heuristic, we would come even closer to the goal of fully unsupervised raw text parsing.

Sentence identification and tokenization Another open issue is the requirement for perfect tokenization and sentence identification. This may seem a relatively minor issue for languages encoded with Latin script (or Greek or Cyrillic), where breaking sentences by punctuation symbols and tokenizing by white-space work reasonably well (though, see Kiss and Strunk (2006) for an unsupervised method to learn good sentence identification). The issue is different for Chinese, as well as many of the world's scripts, which do not use white-space to identify word segmentation. Several unsupervised approaches approaches exist for Chinese word segmentation, see Zhao and Kit (2008) for a survey, as do some for Japanese and Korean. But for many languages there is little research on this problem – and these

are precisely the languages for which we have few or no annotated resources in general, where unsupervised methods may be usefully applied.

Issues with the PRLG As noted, the PRLG model performs well for the datasets discussed in this dissertation, but may not generalize to typologically different languages as well as the HMM. In short, the PRLG is not a *reversible* model, and as such seems to favor languages – like English – which often feature high-frequency function words at the start of constituents or phrases.

Summing up, this research shows that a simple method can sometimes make progress on a difficult problem. This is a positive result for our field, not only for the progress on the problem itself, but the opportunities for further progress it enables. One of my goals for this research is that the simplicity and extensibility of these methods – and the open-source implementation I provided – will encourage other researchers to take up these ideas, improve them, and, hopefully, apply them in ways I have not yet envisioned. If any do, I wish them the best of luck.

Bibliography

S. Abney. *The English noun phrase in its sentential aspect*. PhD thesis, MIT, 1987.

S. Abney. Parsing by chunks. In R. Berwick, S. Abney, and C. Tenny, editors, *Principle-based Parsing*. Kluwer, 1991.

P. W. Adriaans, M. Trautwein, and M. Vervoort. Towards high speed grammar induction on large text corpora. In *SOFSEM*, 2000.

M. Bendersky, W. B. Croft, and D. A. Smith. Two-stage query segmentation for information retrieval. In *Proc of SIGIR*, 2009.

T. Berg-Kirkpatrick, A. Bouchard-Côté, J. DeNero, and D. Klein. Painless unsupervised learning with features. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 582–590, Los Angeles, California, June 2010. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/N10-1083`.

S. Bergsma and D. Lin. Bootstrapping path-based pronoun resolution. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 33–40, Sydney, Australia, July 2006. Association for Computational Linguistics. doi: 10.3115/1220175.1220180. URL `http://www.aclweb.org/anthology/P06-1005`.

C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

E. Black. Meeting of the interest group on evaluation of broad-coverage grammars of English, 1992. URL `http://linguistlist.org/issues/3/3-587.html`.

L. Bloomfield. *Language*. Henry Holt, 1933.

P. Blunsom and T. Cohn. Unsupervised induction of tree substitution grammars for dependency parsing. In *Proceedings of the 2010 Conference on Empirical Methods in*

Natural Language Processing, pages 1204–1213, Cambridge, MA, October 2010. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D10-1117.

P. Blunsom and T. Cohn. A hierarchical pitman-yor process hmm for unsupervised part of speech induction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 865–874, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P11-1087.

R. Bod. An all-subtrees approach to unsupervised parsing. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 865–872, Sydney, Australia, July 2006a. Association for Computational Linguistics. doi: 10.3115/1220175.1220284. URL http://www.aclweb.org/anthology/P06-1109.

R. Bod. Unsupervised parsing with U-DOP. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 85–92, New York City, June 2006b. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W06/W06-2912.

T. Brants. Cascaded markov models. In *Ninth Conference of the European Chapter of the Association for Computational Linguistics*, 1999. URL http://aclweb.org/anthology/E/E99/E99-1016.pdf.

T. Brants. Inter-annotator agreement for a German newspaper corpus. In *Second International Conference on Language Resources and Evaluation LREC-2000*, Athens, Greece, 2000.

D. J. Brooks. Unsupervised grammar induction by distribution and attachment. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 117–124, New York City, June 2006. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W06/W06-2916.

P. F. Brown, P. V. deSouza, R. L. Mercer, V. J. Della Pietra, and J. C. Lai. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4), December 1992.

C. Carroll and E. Charniak. Two experiments in learning probablistic dependency grammars from corpora. In C. Weir, S. Abney, R. Grishman, and R. Weischedel, editors, *Working Notes of the Workshop Statistically-Based NLP Techniques*, pages 1–13. AAAI Press, 1993.

N. Chomsky. *Syntactic Structures*. Mouton, 1957.

N. Chomsky. *Aspects of a Theory of Syntax*. MIT Press, 1965.

N. Chomsky. *Knowledge of Language: Its nature, origin, and use*. Greenwood Publishing Group, 1986.

C. Christodoulopoulos, S. Goldwater, and M. Steedman. Two decades of unsupervised POS induction: How far have we come? In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 575–584, Cambridge, MA, October 2010. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D10-1056.

S. B. Cohen and N. A. Smith. Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 74–82, Boulder, Colorado, June 2009. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/N/N09/N09-1009.

S. B. Cohen, K. Gimpel, and N. A. Smith. Logistic normal priors for unsupervised probabilistic grammar induction. In *NIPS*, 2009.

M. Collins. *Head-Driven Statistical Models for Natural Language Parsing*. PhD thesis, University of Pennsylvania, 1999.

N. Constant, C. Davis, C. Potts, and F. Schwarz. The pragmatics of expressive content: Evidence from large corpora. *Sprache und Datenverarbeitung*, 33(1–2): 5–21, 2009.

B. Cramer. Limitations of current grammar induction algorithms. In *Proceedings of the ACL 2007 Student Research Workshop*, pages 43–48, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P/P07/P07-3008.

D. Davidov, R. Reichart, and A. Rappoport. Superior and efficient fully unsupervised pattern-based concept acquisition using an unsupervised parser. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, pages 48–56, Boulder, Colorado, June 2009. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W09-1108.

X. Duan, J. Zhao, and B. Xu. Probabilistic models for action-based Chinese dependency parsing. In *ECML/PKDD*, 2007.

J. Eisenstein, B. O'Connor, N. A. Smith, and E. P. Xing. A latent variable model for geographic lexical variation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1277–1287, Cambridge, MA, October 2010. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D10-1124.

W. N. Francis and H. Kucera. *Brown Corpus Manual*, 1964. URL http://khnt.aksis.uib.no/icame/manuals/brown/. Revised 1971, revised and updated 1979.

K. Gimpel, N. Schneider, B. O'Connor, D. Das, D. Mills, J. Eisenstein, M. Heilman, D. Yogatama, J. Flanigan, and N. A. Smith. Part-of-speech tagging for Twitter: Annotation, features, and experiments. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics, companion volume*, Portland, OR, June 2011.

T. L. Griffiths, M. Steyvers, D. M. Blei, and J. M. Tenenbaum. Integrating topics and syntax. In *NIPS*, 2005.

C. Hänig. Improvements in unsupervised co-occurence based parsing. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 1–8, Uppsala, Sweden, July 2010. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W10-2901.

W. P. Headden III, D. McClosky, and E. Charniak. Evaluating unsupervised part-of-speech tagging for grammar induction. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 329–336, Manchester, UK, August 2008. Coling 2008 Organizing Committee. URL http://www.aclweb.org/anthology/C08-1042.

W. P. Headden III, M. Johnson, and D. McClosky. Improving unsupervised dependency parsing with richer contexts and smoothing. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 101–109, Boulder, Colorado, June 2009. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/N/N09/N09-1012.

I. Heim. *The semantics of definite and indefinite noun phrases*. PhD thesis, University of Massachusetts, 1982.

K. Hollingshead, S. Fisher, and B. Roark. Comparing and combining finite-state and context-free parsers. In *Proceedings of Human Language Technology Conference*

and Conference on Empirical Methods in Natural Language Processing, pages 787–794, Vancouver, British Columbia, Canada, October 2005. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/H/H05/H05-1099.

F. Jelinek. *Statistical Models for Speech Recognition*. MIT Press, 1998.

A. K. Joshi. Hierarchical structure and sentence description. In B. E. Nevin and S. M. Johnson, editors, *The Legacy of Zellig Harris: Language and information into the 21st Century, Volumn 2: Mathematics and computability of language*. John Benjamins, 2002.

A. K. Joshi and P. Hopely. A parser from antiquity. *Natural Language Engineering*, 2(04):291–294, 1996. doi: 10.1017/S1351324997001538. URL http://dx.doi.org/10.1017/S1351324997001538.

M. Joshi, D. Das, K. Gimpel, and N. A. Smith. Movie reviews and revenues: An experiment in text regression. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 293–296, Los Angeles, California, June 2010. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/N10-1038.

D. Jurafsky and J. H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Processing, 2nd ed.* Pearson, 2008.

H. Kamp and E. Reyle. *From Discourse to Logic: Introduction to model theoretic semantics of natural language, formal logic and discourse representation theory*. Kluwer Academic Publishers, 1993.

T. Kiss and J. Strunk. Unsupervised multilingual sentence boundary detection. *Computational Linguistics*, 32(4):485–525, 2006.

D. Klein. *The Unsupervised Learning of Natural Language Structure*. PhD thesis, Stanford, 2005.

D. Klein and C. D. Manning. A generative constituent-context model for improved grammar induction. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 128–135, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073106. URL http://www.aclweb.org/anthology/P02-1017.

D. Klein and C. D. Manning. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42nd Meeting of*

the Association for Computational Linguistics (ACL'04), Main Volume, pages 478–485, Barcelona, Spain, July 2004. doi: 10.3115/1218955.1219016. URL http://www.aclweb.org/anthology/P04-1061.

B. Krenn, T. Brants, W. Skut, and Hans Uszkoreit. A linguistically interpreted corpus of German newspaper text. In *Proceedings of the ESSLLI Workshop on Recent Advances in Corpus Annotation*, 1998.

K. Lari and S. J. Young. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech & Language*, 4:35 – 56, 1990.

R. Levy and C. D. Manning. Is it harder to parse chinese, or the chinese treebank? In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 439–446, Sapporo, Japan, July 2003. Association for Computational Linguistics. doi: 10.3115/1075096.1075152. URL http://www.aclweb.org/anthology/P03-1056.

J. Lin and C. Dyer. *Data-Intensive Text Processing with MapReduce*. Morgan & Claypool, 2010.

C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.

M. P. Marcus, B. Santorini, M. A. Marcinkiewicz, and A. Taylor. *Treebank-3*. Linguistic Data Corporation, 1999.

M.P. Marcus, B. Santorini, and M.A. Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. *Compuational Linguistics*, 19(2):313–330, June 1993.

D. McClosky, E. Charniak, and M. Johnson. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 152–159, New York City, USA, June 2006. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/N/N06/N06-1020.

I. Mel'čuk. *Dependency Syntax: Theory and Practice*. SUNY Press, 1988.

T. Moon, J. Baldridge, and K. Erk. Crouching Dirichlet, hidden Markov model: Unsupervised POS tagging with context local tag generation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 196–206, Cambridge, MA, October 2010. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D10-1020.

M. Moortgat. Categorial type logics. In J. van Benthem and A. T. Meulen, editors, *Handbook of Logic and Language*. MIT Press, 1997.

M. Palmer, F. D. Chiou, N. Xue, and T. K. Lee. *Chinese Treebank 5.0*. Linguistic Data Corporation, 2005.

F. Pereira and Y. Schabes. Inside-outside reestimation from paritally bracketed corpora. In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics*, pages 128–135, Newark, Delaware, USA, June 1992. Association for Computational Linguistics. doi: 10.3115/981967.981984. URL `http://www.aclweb.org/anthology/P92-1017`.

E. Ponvert, J. Baldridge, and K. Erk. Simple unsupervised prediction of low-level constituents. In *Proceedings of the Fourth IEEE International Conference on Semantic Computing*, Pittsburg, PA, 2010.

E. Ponvert, J. Baldridge, and K. Erk. Simple unsupervised grammar induction from raw text with cascaded finite state models. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, Portland, OR, 2011.

V. Punyakanok, D. Roth, and W. Yih. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics: Special Issue on Semantic Role Labeling*, 34(2), June 2008.

L.R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 1989.

A. Radford. *Transformational Grammar: A first course*. Cambridge University Press, 1988.

A. Radford. *Minimalist syntax: Exploring the structure of English*. Cambridge University Press, 2004.

L. A. Ramshaw and M. P. Marcus. Text chunking using transformation-based learning. In *Third Workshop on Very Large Corpora*, 1995.

L. Ratinov and D. Roth. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, pages 147–155, Boulder, Colorado, June 2009. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W09-1119`.

Sujith Ravi and Kevin Knight. Minimized models for unsupervised part-of-speech tagging. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 504–512, Suntec, Singapore, August 2009. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P/P09/P09-1057`.

R. Reichart and A. Rappoport. Unsupervised induction of labeled parse trees by clustering with syntactic features. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 721–728, Manchester, UK, August 2008. Coling 2008 Organizing Committee. URL http://www.aclweb.org/anthology/C08-1091.

R. Reichart and A. Rappoport. Improved fully unsupervised parsing with zoomed learning. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 684–693, Cambridge, MA, October 2010. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D10-1067.

W. Schuler, S. AbdelRahman, T. Miller, and L. Schwartz. Broad-coverage parsing using human-like memory constraints. *Compuational Linguistics*, 3(1), 2010.

Y. Seginer. Fast unsupervised incremental parsing. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 384–391, Prague, Czech Republic, June 2007a. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P07-1049.

Y. Seginer. *Learning Syntactic Structure*. PhD thesis, University of Amsterdam, 2007b.

F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, 2003. URL http://aclweb.org/anthology/N/N03/N03-1028.pdf.

N. A. Smith and J. Eisner. Annealing techniques for unsupervised statistical language learning. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 486–493, Barcelona, Spain, July 2004. doi: 10.3115/1218955.1219017. URL http://www.aclweb.org/anthology/P04-1062.

N .A. Smith and J. Eisner. Annealing structural bias in multilingual weighted grammar induction. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 569–576, Sydney, Australia, July 2006. Association for Computational Linguistics. doi: 10.3115/1220175.1220247. URL http://www.aclweb.org/anthology/P06-1072.

N. A. Smith and M. Johnson. Weighted and probabilistic CFGs. *Computational Lingusitics*, 2007.

B. Snyder, T. Naseem, and R. Barzilay. Unsupervised multilingual grammar induction. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 73–81, Suntec, Singapore, August 2009. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P/P09/P09-1009.

Z. Solan, D. Horn, E. Ruppin, and S. Edelman. Unsupervised learning of natural languages. *PNAS*, 102, 2005.

V. I. Spitkovsky, H. Alshawi, and D. Jurafsky. From baby steps to leapfrog: How "less is more" in unsupervised dependency parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 751–759, Los Angeles, California, June 2010a. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/N10-1116.

V. I. Spitkovsky, H. Alshawi, D. Jurafsky, and C. D. Manning. Viterbi training improves unsupervised dependency parsing. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 9–17, Uppsala, Sweden, July 2010b. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W10-2902.

V. I. Spitkovsky, D. Jurafsky, and H. Alshawi. Profiting from mark-up: Hyper-text annotations for guided parsing. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1278–1287, Uppsala, Sweden, July 2010c. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P10-1130.

V. I. Spitkovsky, H. Alshawi, and D. Jurafsky. Punctuation: Making a point in unsupervised dependency parsing. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, pages 19–28, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W11/W11-0303.

M. Steedman. *Surface Structure and Interpretation*. MIT Press, 1996.

M. Steedman. *The Syntactic Process (Language, Speech and Communication)*. MIT Press, 2001.

L. Tesnière. *Éléments de syntaxe structurale*. Editions Klincksieck, 1959.

E. F. Tjong Kim Sang and S. Buchholz. Introduction to the CoNLL-2000 Shared Task: Chunking. In *Proceedings of the Fourth Conference on Computational Natural Language Learning and of the Second Learning Language in Logic Workshop*, 2000.

E. F. Tjong Kim Sang and F. De Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In Walter Daelemans and Miles Osborne, editors, *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147, 2003. URL http://www.aclweb.org/anthology/W03-0419.pdf.

M. van Zaanen. ABL: Alignment-based learning. In *Proceedings of COLING 2000 Volume 2: The 18th International Conference on Computational Linguistics*, 2000.

B. Webber. Genre distinctions for discourse in the Penn TreeBank. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 674–682, Suntec, Singapore, August 2009. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P/P09/P09-1076.

B. Wing and J. Baldridge. Simple supervised document geolocation with geodesic grids. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 955–964, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P11-1096.

F. Xia. *The Part-Of-Speech Tagging Guidelines for the Penn Chinese Treebank (3.0)*, October 2000. URL http://www.cis.upenn.edu/~chinese/posguide.3rd.ch.pdf.

K. Yamada and K. Knight. A syntax-based statistical translation model. In *Proceedings of 39th Annual Meeting of the Association for Computational Linguistics*, pages 523–530, Toulouse, France, July 2001. Association for Computational Linguistics. doi: 10.3115/1073012.1073079. URL http://www.aclweb.org/anthology/P01-1067.

H. Zhao and C. Kit. An empirical comparison of goodness measures for unsupervised Chinese word segmentation with a unified framework. In *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-I*, 2008. URL http://aclweb.org/anthology/I/I08/I08-1002.pdf.

Vita

Elias Franchot Ponvert was born in 1978 in Newport, Rhode Island, to Antonio Ponvert and Annie Duncan Ponvert. He attended St. Michael's School and then the New School for kindergarten through 8th grade, followed by the Portsmouth Abbey School, graduating in 1996 with a high school diploma. He served a whale spotter for the Frances Fleet whale watching tours out of Narragansett, RI, in 1993 and as a camp counselor at the Episcopal Conference Center in Pascoag, RI from 1994 to 1996. He attended McGill University in Montréal, graduating with a BA in linguistics in 2000. In July 2000 he moved to Austin, Texas to take a position as a computational linguist at Cycorp, Inc. From 2001 to 2006 he was a systems analyst for the Center for Instructional Technologies at the University of Texas at Austin. In 2005 he married Candace Pruett, an artist and model originally from San Antonio; and in that year he began a PhD program in linguistics at the University of Texas, earning an MA in 2008. In July 2008 Candace and Elias had a daughter, Katharine Agnes, and in March 2010 a son, Jasper Nathaniel. To his knowledge, he has not missed a Jonathan Richman show in his city of residence since 1996.

Permanent Address: 3803C Byron Dr
Austin, TX 78704
USA

This dissertation was typeset with LaTeX 2$_\varepsilon$[3] by the author.

[3] LaTeX 2$_\varepsilon$ is an extension of LaTeX. LaTeX is a collection of macros for TeX. TeX is a trademark of the American Mathematical Society. The macros used in formatting this dissertation were written by Dinesh Das, Department of Computer Sciences, The University of Texas at Austin, and extended by Bert Kay, James A. Bednar, Ayman El-Khashab and the author.